Universidad Carlos III de Madrid
Escuela Politécnica Superior

WHOLE ORGAN BIOENGINEERING: DESIGN AND AUTOMATION OF A RECELLULARIZATION BIOREACTOR

Bachelor Thesis
Biomedical Engineering

Author: Daniela Rubio Soto
Tutor: Pedro Baptista Almeida de Matos
Director: Juan Francisco del Cañizo López

Madrid, September 2017

AKNOWLEDGMENTS

I wish to acknowledge all the people that have contributed to the development of my Bachelor Thesis. First, Dr. Sara Guerrero Aspizua and all the education personnel of the University Carlos III because without their help, dedication and attention, this project wouldn't be a reality.

I also want to express my gratitude to Dr. Juan Francisco del Cañizo for his time, for his help and dedication. Him and all the personnel working at the Laboratory of Artificial Circulation at the University Hospital Gregorio Marañón.

Last but no least, to Dr. Pedro Baptista Almeida de Matos for trusting on me since the beginning and showing me that my professional future is in the field of Tissue Engineering and Regenerative Medicine. To him and his entire group at the Instituto de Investigación Sanitaria de Aragon. For all what they taught me since my professional internship, their time, and dedication and especially for making me feel like a member of their group since the beginning.

ABSTRACT

Organ transplantation has saved millions of lives because it is currently the only permanent solution for several diseases. Nevertheless, there are huge drawbacks to overcome such as organ shortage, immune rejection and the need of whole-life immunosupressive regimens. There have been different attempts to solve this problem as 3D organ printing, hepatocyte transplantation, artificial and bioartificial organs. However, it has been shown that Tissue Engineering and Regenerative Medicine are the real solution and have become the future of organ transplantation.

In order to obtain functional bioengineered organs, bioreactors need to be used to automatize and optimize decellularization and recellularization processes. This Bachelor Thesis is intended to design, program and build some of the components of a liver recellularization bioreactor. Specifically, five syringe pumps, two peristaltic pumps, a pH-meter, a level and a weight sensor will be developed. Moreover, this thesis belongs to a bigger project whose main objective is the generation of functional livers that could be transplanted into human patients.

Key words: Tissue Engineering, Regenerative Medicine, bioengineering, bioreactor, decellularization, recellularization, perfusion, scaffold, extracellular matrix and Guided User Interface.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. Introduction

1.1 Liver transplantation: brief history

According to the World Health Organization, a transplant is "*the transfer of human cells, tissues or organs from a donor to a recipient with the aim of restoring its functions in the body*" .[1] Although its history started some centuries ago, it still presents different challenges and limitations such as organ shortage, oxygen and nutrient transport, immune rejection and whole-life immunosuppression requirements. However, it is today the only definitive solution for severe diseases as liver failure.

At the beginning of the twentieth century, the first successful autotransplant of a canine kidney was performed initiating the transplant era. In 1902, Carrel described the triangulation method used in vascular anastomosis.[2] However, it was not his only contribution to this field. Along the 1930s Carrel, who is considered the father of modern regenerative medicine and cardiovascular surgery, developed the perfusion pump together with the engineer Charles Lindbergh. It was an essential technological advance as the machine made it possible to maintain living organs out of the body during surgical procedures. Both of them published most of their ideas in 1938 under the tittle "*The culture of organs*". Carrel and Lindbergh worked together in the Rockefeller Institute in New York in the field of the culture of organs.[3]

It was already in 1954 when the first organ, a kidney, was successfully transplanted. In fact, the procedure took place between identical twins by Murray. From that moment on, more organs and tissues were transplanted, such as bone marrow (1956), heart (1967), liver (1967) and pancreas (1989).

Year	Event
1818	First interhuman transfusion by Blundell
1885	Roux and Harrison establish the principle of tissue culture
1902	Carrel describes the triangulation method for vascular anastomosis
1905	First successful cornea transplant by Zirm
1906	Jaboulay attempts renal xenotransplantation in 2 patients receiving organs from pig and goat
1930s	Carrel and Lindbergh develop the perfusion pump
1954	First successful organ (kidney) transplant between identical twins by Murray in Boston
1967	First successful liver transplant by Starzl
1993	First baboon-to-human liver transplantation by Starzl and coworkers
2002	First successful bioengineered urinary bladder by Atala
2008	First bioengineered trachea transplant in Barcelona by a multidisciplinary team

Table 1: Main events in the history of organ transplantation and regenerative medicine. *Source: Chun S, Lim G, Kwon T et al. Identification and characterization of bioactive factors in bladder submucosa matrix. Biomaterials. 2007;28(29):4251-4256. doi:10.1016/j.biomaterials.2007.05.020.*

The history of liver transplantation itself started in 1955 when Welch described scientifically the process as a treatment. Already in 1958 orthotopic liver transplantation was described by Francis Moore in dogs. Later on, the 1st of March 1963 Starzl performed the first liver transplant. The patient was a boy of three years old who suffered biliary atresia. Nevertheless the child did not survive to the surgery as he died while it was being performed due to an uncontrolled bleeding and coagulation disorder. [5]

None of the first five patients who were liver transplanted survived more than 23 days. It was clear that there were huge difficulties and challenges to overcome in the field. Starzl continued working on the area to develop procedures and principles that allowed liver transplantation to become a clinical reality and that are still used to date.

It was not until 1967 when the first successful liver transplants were performed, by Starzl at the University of Colorado. However, non-selective immunosuppression treatments were given to patients making them vulnerable to infections and increasing their mortality after transplantation. The leading causes of death were infection complications and rejection.

Twelve years later, in 1979, Calne used cyclosporine, an immunosuppressor drug, in two patients after liver transplantation. This surgical procedure was approved by the National Institute of Health in 1983 as a therapy for end-stage liver failure. In 1989 patient survival during the first five years after transplantation increased up to 64%.

Today, fifty years after the first successful liver transplant, more than one hundred thousand liver transplants are performed each year around the world (http://www.transplant-observatory.org). Moreover, survival has considerably increased up to an 80-90% in the first year. Yet, the challenges mentioned before are still present. *"Organ transplantation is a victim of its own success"*. [6]

New strategies are required to solve those limitations. Regenerative medicine and Tissue Engineering pursue ultimately the same objective as organ transplantation. In fact, they are considered to be the future of organ transplantation as they could solve most of its problems, such as nutrient and oxygen transport or rejection.

During 1988 Vacanti developed one of the first experimental studies in the field of Tissue Engineering. This study investigated fetal and adult rat and mouse hepatocytes seeded in synthetic polymeric scaffolds together with pancreatic islet cells and cells from the small intestine. One of the most important breakthroughs in this field occurred in Massachusetts by the 1990s, when skin substitutes were commercialized representing one of the first attempts to engineer tissues.[2] In 2006, Atala reported the first human transplant of a bioengineered organ, autologous urinary bladders for patients needing cytoplasty.[7]

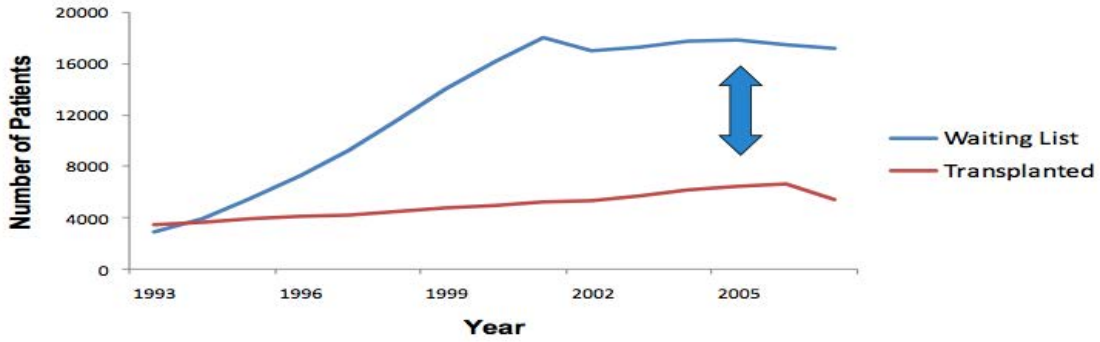

Figure 1: Graph representing the widening of the gap between the transplanted patients and those ones in the waiting list over time. *Source: The Organ Procurement and Transplantation Network.*

1.2 Motivation

Liver transplantation is the only solution for severe diseases affecting this solid organ and it is considered the gold-standard of treatment. It is the case of inborn metabolic disorders, cirrhosis, and injuries from alcohol abuse, chronic viral hepatitis, systemic autoimmune diseases and hepatocellular carcinoma.[8, 9, 10] From the information already presented, it is clear that there is a necessity to increase the number of organs available for transplantation and overcome other challenges in the field.

Organ shortage has been reported to be an increasing problem for today's society. The gap between the patients waiting for a transplant and the available organs has become wider (Figure 1). It has been reported that the deficit increases in about 4000 livers per year.[11] Moreover, the mortality of those ones waiting for a transplant has also increased.[6, 12] There are around 27000 deaths annually in the United States of patients suffering end-stage liver failure and waiting for a liver transplant [11, 13]. And this numbers do not even include the patients that are too sick to be considered for transplant.

The number of patients requiring a transplant has increased faster than the number of donated organs. There have been different approaches to solve the shortage by increasing the pool of organs including live donation, donation after cardiac death, split liver technique, chain transplantation and improved models for organ transplantation.[14] Furthermore, technological advances allow better maintenance of the organs before being transplanted. However, all these efforts are still insufficient.

Notwithstanding, organ shortage is not the only challenge to be solved. There are other limitations as rejection of the organ after being transplanted, control of infections during the surgery and genetic matching of donors and host. Once the transplant has been performed the patient needs immunosuppressive drugs for the rest of his life. This fact may present further side effects as hypertension, opportunistic infections, malignancy or impaired wound healing.

1.2.1 Possible solutions to the problem

There are different solutions that could solve the limits of organ transplantation: 3D organ printing, artificial organs, bioartificial organs and hepatocyte transplantation. Three-dimensional organ printing is based on the same technology as 3D printing but it uses biological inks instead of materials as synthetic polymers. On the one hand, this technology has succeeded for simple and avascular tissues as skin.[15] However, considerably more complex technology will be required to obtain functional solid and vascularized organs.[16]

Hepatocyte transplantation could be considered to be the most significant alternative to whole liver transplantation. This approach was first used to cure familial hypercholesterolemia as well as other liver diseases. In spite of this, results are non-convincing. Some of the main disadvantages that limit this strategy are the low efficiency of the engraftment and the low number of cells that can be supplied. The second drawback could be solved with the use of stem cells but there is no clear solution for the first one.[11, 13, 17]

Artificial organs are mechanical systems made of synthetic materials as plastic or metal alloys. They are based on physico-chemical mechanisms used to mimic the functions of the replaced organ. It is the case of LDU (Liver Dialysis Unit) that was approved by the FDA in 1994, which includes activated charcoal in the dialysate. The main disadvantages of these systems include thrombogenicity, unsuccessful endothelialization, no interaction with the organism and lack of adaptation.

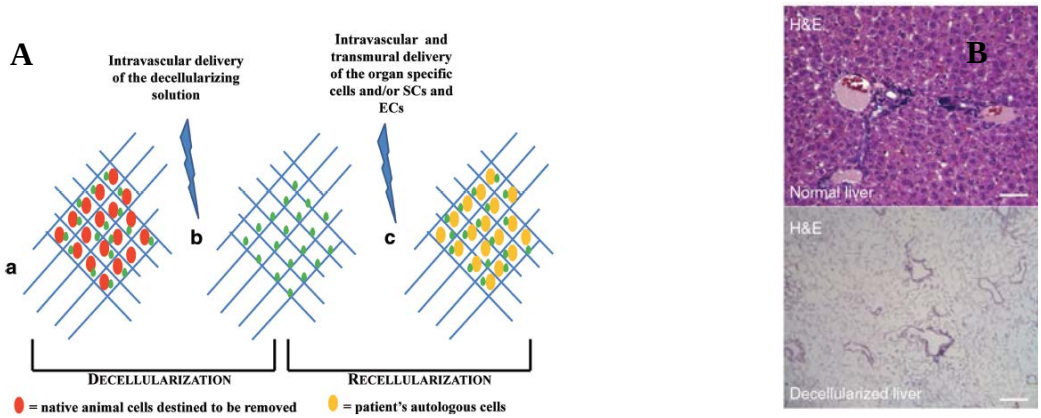

Figure 2: **(A)** Schematic representation of a semi-transplantation based on decellularization and recellularization. *Source: Orlando G, Wood K, Stratta R, Yoo J, Atala A, Soker S. Regenerative Medicine and Organ Transplantation: Past, Present, and Future. Transplantation. 2011;91(12):1310-1317. doi:10.1097/tp.0b013e318219ebb5*. **(B)** Comparison of normal (top) and decellularized (bottom) liver with H&E staining. *Source: Uygun B, Soto-Gutierrez A, Yagi H et al. Organ reengineering through development of a transplantable recellularized liver graft using decellularized liver matrix. Nature Medicine. 2010;16(7):814-820. doi:10.1038/nm.2170.*

Bioartificial organs are based on the mixture of synthetic and biological components used to replace a damaged or failing organ. In this case several considerations have to be considered: animal source, purification of cellular components, infectious diseases transmission, biocompatibility and blood perfusion. Bioartificial livers (BALs) such as ELAD, HepAssist2000 system, Liverx2000 and MELS are not stable substitutes for liver transplant.[13]

1.2.2 Tissue Engineering as a solution

Tissue Engineering and Regenerative Medicine are the solution and future of organ transplantation. In particular bioengineered organs based on decellularization and recellularization processes (Figure 2) may resolve organ shortage by using animal organs as the starting point and autologous cells to prevent rejection and immunosuppression necessity.

The resulting semi-xenotransplantation allows more accurate replication of developmental programs, recreation of the cell niche and development of more complex structures. Even more important, this approach is considered to be a clear success because the organ vasculature remains intact after the decellularization being suitable for organ perfusion and surgical anastomosis.[6, 18] As a result, recellularized organs, once the process becomes optimized and reproducible with the use of a bioreactor, could be transplanted into human patients and be used in drug development.

1.3 Decellularization and Recellularization processes

Bioengineered organs are obtained from a series of processes including organ harvest, decellularization, recellularization and transplantation back into the patient. Additionally, it is also essential to characterize every step of the process. For example, it is required to analyze the decellularized scaffold to ensure no cellular component remains in the structure and that the vasculature has not been damaged.

In order to achieve optimization, automation[19], reproducibility and safety, bioreactors are required. According to the IUPAC, a bioreactor is *"any manufactured or engineered device or system that supports a biologically active environment"* [20]. They are used in Tissue Engineering for cell seeding of 3D scaffolds, maintenance of a controlled culture environment, perfusion of the culture media and containment to ensure sterility.

The bioreactor is used in this case to replicate the cellular microenvironment and to perform sensing analysis for quality control. There is no other method rather than bioreactors that allow biochemical and biophysical parameters as pH to be controlled and adjusted during the entire process. [20]

1.3.1 Liver harvesting: animal source

The starting point of liver bioengineering is the harvest[8] of the organ from pigs or human donors. Using animal source organs had been the objective of many researchers, but rejection and surgical complications caused xeno-transplant failure.[22, 23] However, animal cells will be destroyed and removed during decellularization so that rejection or infections are avoided. Then, the cells that will be used may be autologous, resulting in a semi-xenotransplantation. In some cases, the entire organ will not be necessary and hepatectomy will be performed before decellularization.

There are different reasons to obtain organs from these animals. On the one hand, organ shortage will not be a problem any longer. Furthermore, there will be no microbiological risk because animal cells will not be used and highly controlled herds will be utilized, eliminating the risk of porcine endogenous retrovirus replication.[22]

The use of animal organs may also involve less ethical controversy if the processes take place under a legal framework. Due to cultural taboos, deceased donation is not legal in many countries increasing the organ shortage. This solution will reduce legal, ethical and practical problems. Even more significant, porcine liver scaffolds are similar to human livers in size and the safety has been already demonstrated with the use of other structures such as heart valves. [13]

1.3.2 Cell culture

At the same time that the liver is harvested and decellularized, the cells have to be cultured in vitro. It is a difficult and slow process as millions of cells are required (the exact number depends on the volume of the scaffold) and they may be highly sensitive. Moreover, it becomes more difficult if the cells used are stem cells, as their fate will have to be tightly controlled.

At the end, all types of cells present in native livers will have to be cultured to be seeded on the scaffolds. It is the case of hepatocytes, endothelial cells and Kupffer cells. Likewise, cell seeding will have to be performed in the appropriate order so that cells can migrate to their native location in the scaffold.

1.3.3 Decellularization process

The ECM (extracellular matrix) has to be tightly preserved as it will provide the vasculature of the bioengineered organ. Moreover, the ECM is the structural support of cells, allows migration and proliferation and it regulates cellular signaling as it contains different bioactive molecules. Then, the decellularization process will have to be highly controlled and refined for each organ to improve results.[21]

Simple decellularization protocols have been designed to remove the cellular components while the vasculature remains intact. There are two main

types of decellularization methods. On the one hand, they can be physical. It is the case of freezing and thawing cycles used to disrupt cells reducing the use of chemical agents. Another physical agent may be pressure. However, it may damage the scaffold and destroy the vasculature.

On the other hand, chemical and enzymatic methods are also used. In that case pH changes, alkaline and acid solutions, non-ionic detergents (Triton), ionic detergents (SDS), alcohols or other enzymatic agents. The most common solutions used are based on detergents. It has been demonstrated[8] that Triton is strong enough to remove the cellular components preserving the ECM and the vasculature. In fact, it could be considered to be the most common agent used in decellularization protocols.

Decellularization process does not require the sterility needed for recellularization. However, control of some parameters as DNA content is also required. Then, bioreactors are also used during this process to improve and automatize results. Decellularized organs are the starting point in recellularization. Then, techniques to remove cellular contents of organs have to be improved to ensure that the best result will be obtained.

Baptista et al. described in 2009[6] an efficient protocol to develop decellularization. In this case the decellularization solution included Triton to break the cellular membrane and deionized water to produce cellular lysis and remove possible toxic detergent residues. In the case of liver decellularization, the resulting scaffold becomes transparent (Figure 3) allowing even the visualization of the vasculature with the naked eye.

1.3.4 Characterization of the decellularization process

Without an appropriate scaffold further organ bioengineering will not be possible, even if the best recellularization techniques were available. As a consequence, exhaustive characterization of the scaffold will be necessary to improve the decellularization process.

Some of the most common techniques used include hematoxilin and eosin staining to visualize cellular components, immunohistochemical analysis, biochemical analysis, visualization under the microscope, tensile test, MRI and CT. DNA quantification is a crucial test required to verify the efficiency of removal of cellular components. Different studies have shown that up to a 98% of cellular DNA can be removed with simple decellularization methods[6].

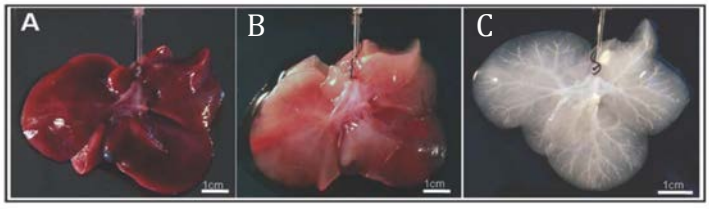

Figure 3: Macroscopic view of a rat liver after (A) 0 min, (B) 20 min and (C) 120 min of decellularization process. *Source: Baptista P, Siddiqui M, Lozier G, Rodriguez S, Atala A, Soker S. The use of whole organ decellularization for the generation of a vascularized liver organoid. Hepatology. 2011;53(2):604-617. doi:10.1002/hep.24067.*

Furthermore, the presence of the most relevant ECM proteins has to be also verified. It is the case of collagen, elastin and fibronectin. Different studies showed that those proteins remain in the scaffolds without being altered and without changing their location. Analysis based on the use of antibodies against those proteins revealed that the matrix chemistry was not altered. There are also simpler complementary studies to confirm the integrity of the vascular network as fluid perfusion.[6]

1.3.5 Recellularization process

Recellularization process is based on the repopulation of a decellularized bioscaffold. It is in fact the last step before transplanting the bioengineered organ back to the patient. Contrary to the decellularization process, recellularization requires complete sterility to avoid bacterial/fungal/viral contamination, as cellular components will be directly exposed to the human body of the transplanted patient.

This process could take place in vivo or in vitro. However, studies have shown that results improve when it takes place in vitro with the use of bioreactor technology[8]. These devices are used to mimic the signals that the liver receives from the body during its development. To do so, pH, glucose, temperature and oxygen levels among other parameters have to be exhaustively controlled and adjusted.

Once the scaffold is prepared and the cells have grown, recellularization can take place. To do so the components of the bioreactor need to be sterilized prior to be used. Then, they will be assembled inside a fume hood using sterile gloves and surgical coat. Next, the system will be primed so that culture media is introduced to remove air from the tubing.

When all the components like pumps, sensors or reservoirs are placed in the appropriate location, the cells can be seeded. To do so the best choice is to incorporate a specific port in the bioreactor vessel designed just for cell seeding. In some cases, more than one cellular injection will be required to prevent cellular aggregations. Finally, some time will be required to allow cellular adhesion to the scaffold before the bioreactor can be disassembled. During this time parameters as flow, pressure, pH, glucose, or gases levels will be measured to control and adjust them if required.

1.4 Biological Background

The ultimate purpose of this project is to obtain functional healthy livers to be transplanted. Thus, it is required to understand its anatomy and physiology. Furthermore, histological analysis will be performed in order to verify the absence of cellular components after decellularization and to check cellular organization once recellularization has been performed. As a consequence, histology of the solid organ has to be also known.

1.4.1 Liver anatomy

Technological advances have served to improve the knowledge available about the liver and its anatomy. However, the majority of the information comes from the work of Claude Couinaud, a French surgeon author of *"The liver: Anatomic and Surgical Studies"*. [24]

The liver is a solid organ of high complexity composed of multiple tissues and cellular types. It is the heaviest (1.4 kg on average for an adult) human internal organ and the second largest one after skin. It is located inferior to the diaphragm in the right upper quadrant of the abdominal cavity. Visceral peritoneum and dense connective tissue cover the liver, also protected by the ribs.

It is divided into four lobes of different size and shape. The falciform ligament divides frontally the solid organ into two main lobes, the left lobe and a bigger right one. This ligament also serves to maintain the organ in its position in the abdominal cavity. The other two smaller lobes are named caudate and quadrate. They are placed posteriorly to the right lobe.

Microscopically, each liver lobe is made up of hepatic lobules, considered as the basic functional units. The lobules consist of plates of hepatocytes radiating from a central vein. A distinctive component of a lobule is the portal triad, which consists mainly of a branch of the hepatic artery, a branch of the hepatic portal vein and a bile duct. However, it also includes lymphatic vessels and a branch of the Vagus nerve.

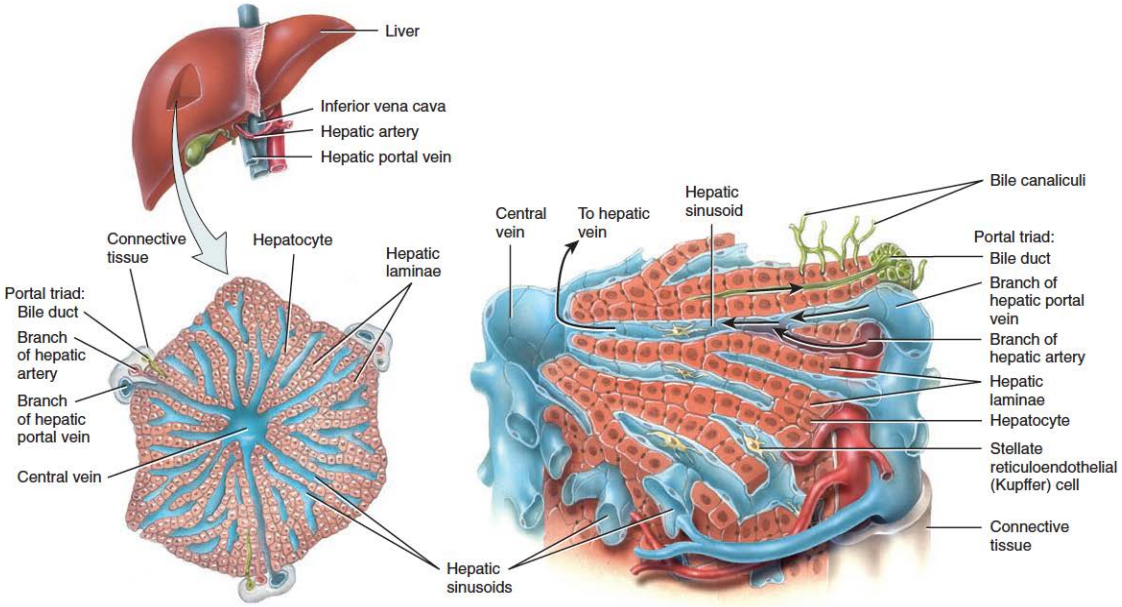

Figure 4: **(A)** Overview of histological components of liver. **(B)** Details of histological components of liver. *Source*: *Tortora G, Derrickson B. Principles Of Anatomy & Physiology. 13th ed. Hoboken, NJ: Wiley; 2012:991.*

1.4.2 Liver physiology

The liver has a wide range of functions including digestion, metabolism, detoxification, storage and production of some compounds and immunity. It secretes bile, which is needed for the absorption of dietary fats and stored in the gallbladder. This organ is also involved in carbohydrate, lipid and protein metabolism. It is also in charge of drugs and hormones processing, excretion of bilirubin derived from the heme degradation of aged red blood cells, synthesis of bile salts, activation of vitamin D and phagocytosis of aged red blood cells, white blood cells and some bacteria.

The liver as a gland is particularly important to maintain blood glucose at normal levels. If the level is below the physiological value, the liver generates glucose through the process of gluconeogenesis and degrades glycogen to glucose, releasing it into the bloodstream by stimulation of glucagon hormone. However, if glucose level is above physiological value the pancreas will secrete insulin stimulating glucose uptake from blood and its storage in the liver as glycogen.

1.4.3 Liver histology

The liver is formed by three major histological components: hepatocytes, bile canaliculi and hepatic sinusoids. Hepatocytes, as the liver functional cells, accomplish secretory, metabolic, and endocrine functions and compose an eighty per cent of the total volume of the organ. These specialized endothelial cells arrange in three-dimensional structures called hepatic laminae, which are bordered by hepatic sinusoids and are characterized by being highly branched and irregular.

There are also small ducts between the hepatocytes called bile canaliculi, which collect the bile. Then, the bile passes to bile ductules and eventually to bile ducts. The last ones, merge and finally give rise to the right and left hepatic ducts. They merge giving rise to the hepatic duct, which forms the bile duct together with the cystic duct from the gallbladder. The bile will ultimately get to the small intestine and contribute to digestion. The direction of bile flow is opposite to the direction of blood, which travels towards a central vein.

The third histological component of the liver is formed by the hepatic sinusoids. These structures are blood capillaries characterized by being highly permeable and being placed between rows of hepatocytes. They receive oxygenated and deoxygenated blood from the hepatic artery and the hepatic portal vein, respectively.

There is a second highly relevant type of cells: Kupffer cells. They are specialized macrophages located in the liver facing the walls of the sinusoids that form part of the reticuloendothelial system, and so they are part of the immune system. These cells are responsible of some functions already mentioned as the phagocytosis of aged red blood cells, white blood cells and some bacteria in venous blood coming from the gastrointestinal tract.

1.4.4 Liver irrigation

The liver receives a dual blood supply: oxygenated blood from the hepatic artery and deoxygenated blood from the hepatic portal vein. The second one delivers about three-quarters of the blood supply to the liver coming from the gastrointestinal tract and the spleen. This blood is rich in nutrients, drugs and some microbes and toxins from the digestive tract.

Blood from both sources flows through the liver sinusoids to allow oxygen, nutrients and some toxins to be absorbed by the hepatocytes. Products synthesized by hepatocytes and nutrients are secreted again into blood, which is drained into the central vein of each lobule and finally into the hepatic veins. The central veins merge into hepatic veins, which leave the liver and drain into the inferior vena cava. [25, 26]

2. Objectives

This Bachelor Thesis belongs to a larger project, as it will be described in section four (General description of the project). The aim of the project is to design, build and program a complete bioreactor to simulate liver developmental environment in order to achieve an optimized, reproducible, automated and controlled recellularization. At the end, the recellularized livers will have to be anatomically, physiologically and histologically identical to native livers so that they can be transplanted ensuring patients survival.

The main objective of this Bachelor Thesis is to complete the second and the third phases of the project. This means that the purpose is to first design the syringe and peristaltic pumps, the level and weight sensor, as well as a pH-meter and then develop the software required to control them in a unique computer program. For that, it is also required to design a suitable graphical user interface, characterized by being user-friendly.

On the other hand, there is also the objective to manufacture all the mechanical components of the pumps and sensors, as well as the electronic elements designed in the second phase. In that way, the result would be the fabrication of five syringe pumps, two peristaltic pumps, a level sensor, a weight sensor and a pH-meter that can be controlled by the user to optimize and automatize the recellularization process.

Finally, additional objectives of the Bachelor Thesis are to document both the hardware and the software and to determine what will be required to complete future phases.

3. State of the art

During the last three decades, Regenerative Medicine and Tissue Engineering have revealed their potential to transfer medicine from the bench to the bedside. It has been possible thanks to the advances made in stem cell biology, sensor technology and materials science.[17] Successful results have been reported for vessels, upper airways, bladders and skin. In 2006 Atala was able to transplant a urinary bladder[7] and two years later Paolo Macchiarini did it for a bioengineered trachea.[27]

Even more important, it has been possible to precisely document the decellularization process for large human-scale organs.[28-30] Current challenges in the field of organ bioengineering are no longer in this phase. The real limitations today belong to the recellularization process and especially to the generation of a suitable vascular network. In order to solve them, bioreactor technology has to be improved.

In 2008 a beating heart was obtained after seeding neonatal cardiac cells on the scaffold of a rat heart.[31] Furthermore, in 2014 the first vascularized bioengineered liver with clinical relevance was transplanted in vivo, being able to be maintained with blood flow during 24 hours.

It is clear that until today efforts have been centered on the development and improvement of decellularization techniques. It is logical if we consider that the results from this process are the starting point for recellularization. Today efforts have to be shifted towards recellularization to finally obtain an optimized process of whole organ bioengineering. Nonetheless, decellularization has to be automated to become reproducible and to be translated into the clinic.

4. General description of the project

This Bachelor Thesis belongs to a larger project that includes another Bachelor Thesis (phase I) and a collaboration between different laboratories. In order to understand its scope, each of its phases will be briefly described. As it was mentioned in the objectives, phases two and three are the ones that belong to this Bachelor Thesis.

4.1 Phase I: Pressure and flow control

Pressure and flow are the most relevant mechanical parameters to be controlled in a recellularization bioreactor[21]. Hence, the first phase of this project was to design and program the peristaltic pumps, pressure and flow sensors involved. The control of flow and pressure is based on a PID controller that provides feedback information. All the elements communicate with a computer, which receives and sends information to control the bioreactor.

Two *Masterflex pumps L/S 7523 and Masterflex L/S 8-channel, 4 roller cartridge pump heads* were used for the portal vein and hepatic artery tubing. Furthermore, the flow meters chosen were *Digiflows EXT1* with the corresponding

Ultrasonic clamp on transducers, from Harvard Apparatus. The pressure transducers *ATP-300* and the *TAM-A* transducer amplifiers were also from Harvard Apparatus.

As the resistance of the system equals the ratio between the pressure drop and the flow, there will be two different modes of control. Because resistance is constant while new elements are not introduced to the system, the user can establish a value for pressure or flow and the other parameter will adapt. This means that the bioreactor will have continuous flow or continuous pressure mode. However, there will be some physiological thresholds for both parameters that will limit the process.

4.2 Phase II: Design and software of syringe and peristaltic pumps, level sensor, weight sensor and pH-meter

The second phase of the project has two different steps. First, the components were designed and later on they were programmed. A total of five syringe pumps, two peristaltic pumps, a pH-meter, a weight and a level sensor were designed and programmed along this stage.

Two of the syringe pumps are used to introduce acid or basic solution into the culture media to adjust the pH measured with the pH-meter. The other three syringe pumps are employed to inject insulin, heparin and growth factors, respectively, according to the circadian cycle of the seeded cells.

On the other hand, one of the peristaltic pumps is used to inject culture media when its level is below an established threshold, which will be measured with the level sensor. As a bioreactor has to mimic the internal environment in which the liver develops, it is required to include a peristaltic pump and a hemofilter to simulate the kidneys. Finally, there will be also a weight sensor to measure the liquid in the urine bag of the hemofiltration system.

4.3 Phase III: Fabrication of components from phase II

The objective of this phase is the fabrication of all the components designed and programmed in the previous phase. To do so different mechanical and electronic elements were built and assembled. It will be further described in detail in section five (Materials and Methods).

4.4 Phase IV: Integration of the components from phase II in the recellularization process

Experimental and biological data is required in order to integrate the components of the second phase in the recellularization process. Then, such information as the circadian cycles will be obtained during the fourth phase. Later on, the already finished system will be tested as part of the bioengineering process to check if in that way the result is optimized.

4.5 Phase V: Substitution of the incubator as temperature control and oxygenator

During the previous phases, temperature, pH control (5% CO_2 environment) and oxygenation are performed with an incubator. However, it may not be enough and an alternative will be required. It has to be considered that just the tubing of the hepatic artery should be oxygenated and that cannot be achieved with an incubator. The objective of this phase will be to design, program and fabricate a temperature control system and a mechanism to control the oxygenation of the circuit.

4.6 Phase VI: Integration of the components from phase V in the recellularization process

As it was previously done for the pumps and sensors of phase two, the temperature controller and the oxygenation system will be also integrated into the recellularization process. The purpose of this phase is to check if the substitution of the incubator improves the outcome of the process. To do so, the new components will be incorporated and several recellularizations will be carried out. Depending on the results the system will be maintained or not.

4.7 Phase VII: Metabolic monitoring

The final phase of this project is the incorporation of in-line integrated sensors to measure different parameters during the recellularization and the maintenance stages of the process. Metabolic monitoring is slightly different during both phases. On the one hand, for the first DNA in solution, lactate dehydrogenase (cellular stress marker), glucose, lactate, ions and some markers of hepatocyte damage will have to be monitored. On the other hand, lactate dehydrogenase (cellular stress marker), albumin, glucose, lactate, urea, ions, cholesterol and bile salts will have to be measured for the second one.

5. Materials and methods

The recellularization bioreactor designed, programmed and built in this project will be used to replicate the developmental microenvironment of the liver. Moreover, at the end of the project, it will be also able to monitor relevant parameters to obtain a functional bioengineered liver.

5.1 General description of the recellularization bioreactor

As this device will be used to recellularize liver scaffolds, cell seeding will be performed during the process. There are some aspects to bear in mind about this procedure to prevent undesired outcomes. First, depending on the flow measured in the system, the total volume of culture media being perfused and the total number of cells seeded there will be one or more cellular injections.

Moreover, and in relation with the design of the bioreactor, there will be a cell injection port in the bioreactor vessel. It will be exclusively used to introduce the cells into the system to prevent contamination and to directly inject them into the recipient in which the scaffold will be placed. This port will also include a Luer adapter. Since it will be only opened while the user generates some pressure with the tailored cell-seeding syringe, sterile conditions will be improved.

Although there is no way to ensure that there will be no contamination, there are other things that improve sterility. It has to be considered that all the components of the bioreactor used must be sterilized prior to be used. Then, the materials chosen to fabricate them will have to allow their sterilization with gamma rays, the autoclave or other methods. In addition, both cell seeding and maintenance are carried out in the same vessel, what has been reported to reduce the chance of contamination.[21]

The bioreactor vessel, where the liver scaffold will be located, must have cylindrical shape to prevent corners where non-adherent cells could aggregate. To further improve mixing inside the recipient, the bioreactor vessel has to be placed over a magnetic stirrer (Figure 5). In this way, those cells that have not attached to the scaffold will recirculate improving recellularization efficiency. Related to the bioreactor vessel, the inlet and the outlet should be positioned approximately 120º apart to reduce the risk of bubbles formation due to channeling.

The design of the biorreactor includes a bioreactor vessel and a reservoir. The first one has a cell injection port, two inlets for the hepatic artery and the portal vein and an outlet for the overflow connection. The reservoir includes eight inlets and one outlet. Among the inputs, five of them come from the syringe pumps and the other three correspond to an overflow connection, an additional replacement media entrance and a connection for the hemofiltration system. The outlet connects the reservoir with the bioreactor vessel being divided into two different lines, the portal vein and the hepatic artery.

On the one hand, in the line corresponding to the portal vein, a peristaltic pump, a pressure sensor, a flow sensor and a pulse dampener are included. The pulse dampener is required in this line because the flow must be continuous as it is physiologically. Moreover, it is also used to prevent the formation of bubbles that could destroy the vasculature of the liver.

On the other hand, a pulse dampener is not required in the line of the hepatic artery but flow and pressure sensors are also included. The bioreactor is used to simulate the internal environment of a human body. Then, the signals provided by other organs have to be also portrayed. As a consequence, a hemofiltration system is added in this line. It is based on a low flux hemofilter, a peristaltic pump and a urine bag.

The electrodes of the level sensor (green squares in Figure 5) are placed on the outer face of the reservoir and the probe of the pH-meter will be located inside it. As shown in the figure, the weight sensor is part of the hemofiltration system, being placed after the urine bag.

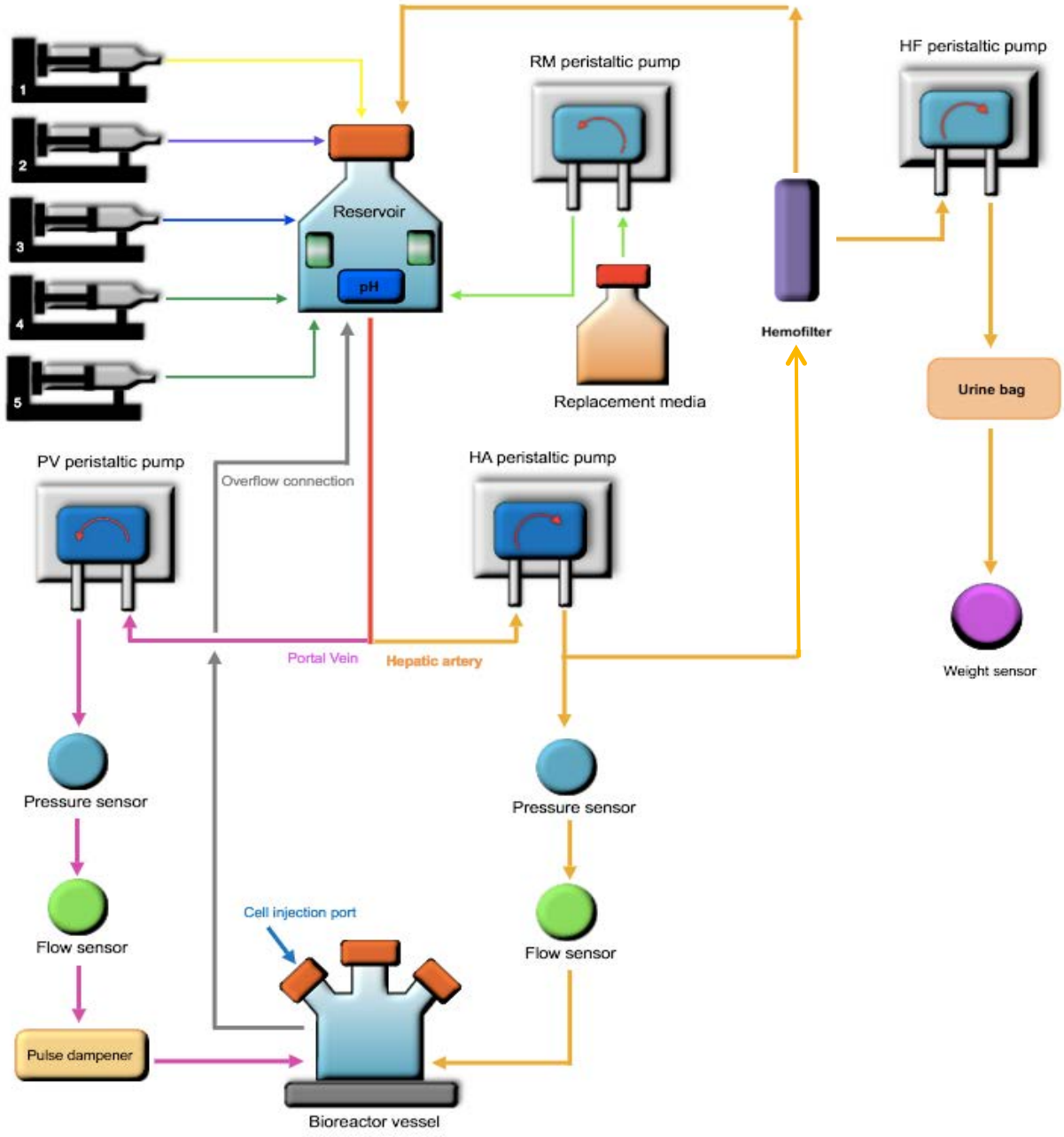

Figure 5: Schematic representation of the recellularization bioreactor including the components of phases one and two: flow and pressure sensors, peristaltic pumps for the portal vein and hepatic artery, five syringe pumps (1: Insulin, 2: Heparin, 3: Growth factors, 4: Base, 5: Acid), replacement media peristaltic pump, hemofilter peristaltic pump, pH-meter, weight and level sensor.

As mentioned above, there are five syringe pumps in the system. The first one is used to inject insulin according to the circadian cycle of the seeded cells. This hormone is included in the system because it is present in the human body to control glucose level in blood, being produced by the pancreas. Then, it stimulates the physiology of the organ. As a consequence, to obtain a functional liver its presence is required. Similarly, glucagon will be used in the fasting periods of the cycle to reproduce the same biological effect.

Heparin is also injected with a syringe pump, the second one, to prevent blood coagulation (if blood is used as a vehicle for red blood cells to work as an oxygenator). The third pump will be used to inject growth factors to stimulate liver development. Although it is not used at this stage of the project, it is already included in the system to provide flexibility for further improvements of the recellularization process. The last two syringe pumps correspond to a basic and an acid solution, respectively, used to adjust the pH measured in the reservoir.

5.2 The Guided User Interface and I^2C

The different elements of the recellularization bioreactor are controlled through a Guided User Interface (GUI) programmed with Gambas3 and different Arduino codes. Gambas is *"a full-featured object language and development environment built on a BASIC interpreter"*.[33] The program created with Gambas3 obtains information from the bioreactor, decodes it so that it can be interpreted and sends new orders back to the system. Gambas3 communicates with the syringe and peristaltic pumps as well as the pH meter, the level and the weight sensors through different Arduino Uno microcontrollers.

The system includes ten Arduino Uno microcontrollers. However, just one of them directly communicates with the computer, the master. The other nine are called slaves and are in charge of controlling the different elements of the system. Initially, the communication with the computer was done directly through those nine microcontrollers using XBee modules, which allow wireless communication. However, there was too much interference what prevented the appropriate functioning of the system. Instead, I^2C serial computer bus is used.

I^2C connections for Arduino Uno	
Pin 5V master	Pull-up resistance 1
Pin 5V master	Pull-up resistance 2
Pull-up resistance 1	Pin A4 master
Pull-up resistance 2	Pin A5 master
Pin 5V master	Pin 5V slave
Pin GND master	Pin GND slave
Pin A4 master	Pin A4 slave
Pin A5 master	Pin A5 slave

Table 2: Connections between Arduino Uno Master and Arduino Uno Slave for I^2C communication.

I^2C bus was developed by Philips in the early 80s for the communication between different components.[34] Its name means Inter-Integrated Circuit. Each slave has its own address and direction so that the master can communicate individually with them. Slaves are not able to start the communication or to transfer information between them. It requires each slave to be connected to the master through the pins A5 and A4 as well as 5V and GND (Table 2).[35] Moreover, pull-up resistances of 2.2k are also included to ensure that the signal is adequate.

Gambas file starts with the initialization of all the variables that are used along the program. The type of variables and the designated names are defined in this section. Next, the program opens the USB port of the computer where the Arduino Uno Master has to be connected. In this case a message indicating an error will be shown if the port cannot be opened. The program also provides the possibility to register the parameters measured in a plaintext. The name of the generated document, the direction where it is saved and its format are predefined along this section. After that the port is read and information is saved as a string.

In addition, each of the buttons from the Guided User Interface has its corresponding section where its function is programmed. All the buttons controlling the same component of the bioreactor belong to the same block (Figure 6). There is an additional block for a chronometer that controls perfusion time and another one for Data Acquisition. Each block is governed by a timer, whose delay determines the time between each repetition of the corresponding code. Finally, the Guided User Interface also contains a security button that stops all the components in case of emergency.

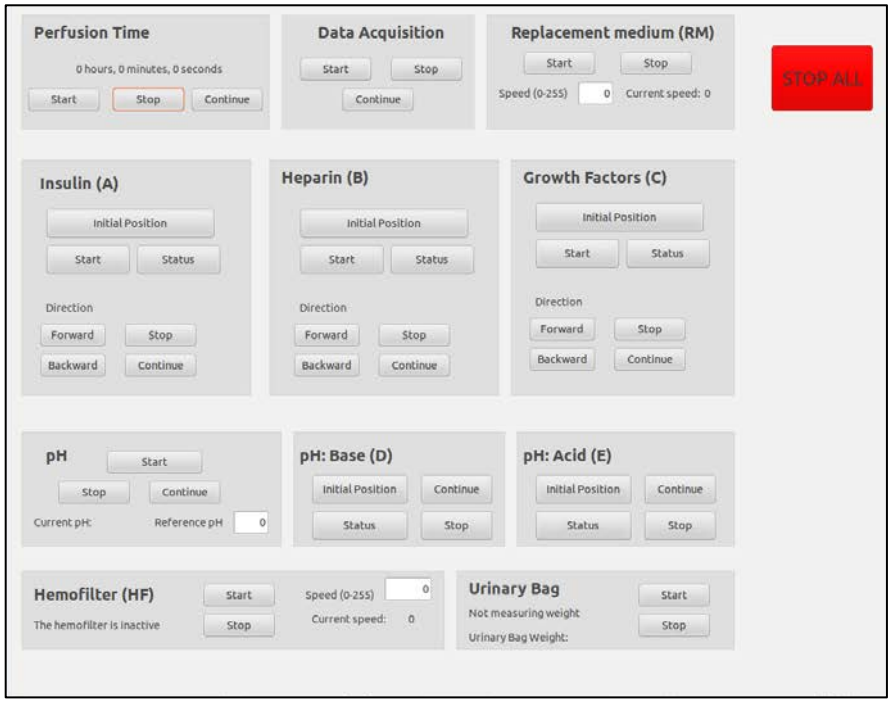

Figure 6: Guided User Interface to control de bioreactor programmed with Gambas3.

When a button is clicked and its function is related with a command, information is sent to the Arduino Uno Master. This microcontroller will decode it to recognize the Arduino Uno Slave to which the information has to be sent. The code of the slave will be run and the order will be executed. There are some commands that will also provide some information back, such as the state of the pumps or the pH measured. Data will go back to the Master and then it will be decoded with Gambas. Finally, parameters will be used to determine the next sequence of events.

5.3 Syringe pumps

The syringe pumps included in this bioreactor were designed based on a model already built by Doctor Juan Francisco del Cañizo López. Their design process, software, electronic circuits and connections are described along this section.

First of all, as in all cases, the pumps were designed. To do so SketchUp software was used. This program provides a three-dimensional representation of the device (Figure 8A) with the advantage of drawing it with size and dimensions information. The syringe pumps are made up of mechanical elements and some pieces that were 3D printed. The mechanical components include a threaded shaft, a shaft adapter, a stepper motor, a stepper motor driver, smooth bars, bearings, a limit switch, profile bars and the external chassis.

The stepper motor chosen is the NEMA17. It divides each rotation into two hundred equal steps. However, the appropriate configuration of the stepper motor driver (TB6560) provides an excitation mode of 1 over 16. At the end, each rotation is equivalent to 3200 steps. This bipolar stepper motor works with 1.2 Amperes and it is characterized by having two different phases and consequently four different cables corresponding to the terminals of each of the two internal coils.

The stepper motor driver has twelve different connections. Four of them are for the motor, other two correspond to the power supply and the last six are connected to the corresponding Arduino Uno Slave. On the one hand, EN-, CW- and CLK- are all connected to each other and to ground from the Arduino Uno Slave. On the other hand, EN+ (enable) connects to Pin 2, CLK+ (number of steps to be moved) to pin 3 and CW+ (direction) to Pin 4.

Figure 7: Picture of one of the syringe pumps already built.

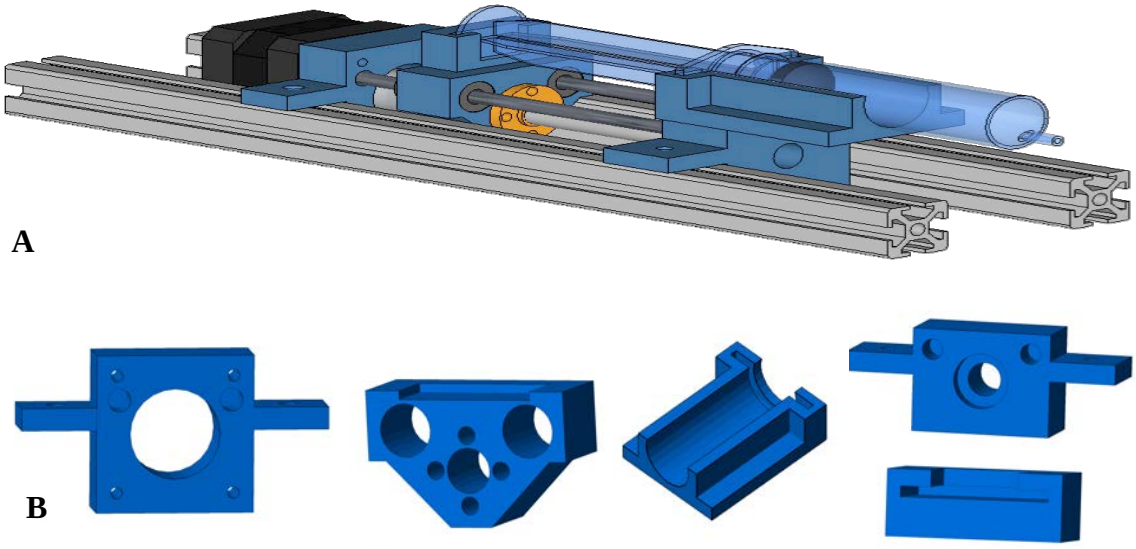

Figure 8: **(A)** Design of the syringe pumps in 3D. **(B)** 3D printed pieces from the syringe pumps.

The limit switch is used to determine the initial position of the syringe in the pump to prevent further movement that could damage the device. It has three different pins: NC or normally closed connected to 5V, NO or normally opened connected to GND and C connected to pin 12 from Arduino Uno Slave. The circuit alternates between the normally closed and normally opened configuration to determine if the syringe is or is not at the initial position.

As mentioned above, some of the components of the syringe pumps were 3D printed (Figure 8B). Once the model of the pump was obtained using SketchUp, the pieces in the printing orientation were exported in STL format to be checked with NetFabb software. Each STL file was transferred to the 3D printer to finally obtain the pieces. They are made up of ABS (acrylonitrile butadiene styrene) as it is easier to manipulate than PLA and it can be used under the required conditions.

The syringe pumps are each one controlled with an Arduino Uno Slave through the stepper motor driver. The most important parameter is the total number of steps required to empty the syringe. In this case, the chosen syringe can be loaded with 20 ml of liquid and it requires moving the piston a total length of 71,5 mm to empty it. The threaded shaft advances 8 mm per revolution and a revolution is equivalent to 3200 steps. Consequently, a revolution is equivalent to 3200 steps or 8 mm. Then, 71,5 mm equals 28600 steps. However, in order to avoid air injection, just 27000 steps will be considered. Moreover, 1430 steps will be equivalent to 1 ml.

The Guided User Interface provides buttons to start, stop and restart the pumps, as well as options to locate them at their initial position and to get status information. Moreover, for insulin, heparin and growth factors additional buttons are available to determine their direction (withdraw or inject). These pumps work based on the current time and the predefined circadian cycles while pumps for basic and acid solutions depend on the pH measured.

Command	Function
CAS	Returns the current state of pump A
CAMxxxxxx	Number of steps that pump A advances
CAD+/CAD-	Defines direction of pump A (- withdraw and + inject)
CAVxxxxxx	Speed in microseconds to advance 1000 steps
CAP	Stops pump A
CAG	Restarts pump A

Table 3: Commands to control syringe pumps. The information is for pump A but can be applied to all pumps (A, B, C, D and E) changing the second letter to the corresponding one (i.e. CBS to obtain the current status of pump B).

When the Arduino Uno Master sends an order to a pump, its Arduino program decodes and interprets it. Table 3 shows the function of the different commands of the syringe pumps. The information shown applies to pump A (insulin), but just changing the second letter of the command to the identifier of the pump (B heparin, C growth factor, D base and E acid) makes it become suitable for the others.

5.4 Peristaltic pumps

This recellularization bioreactor includes four different peristaltic pumps. Two of them (dark blue in Figure 5) were obtained in phase one. However, the other two peristaltic pumps (light blue in Figure 5) have been designed, programmed and built as part of phases two and three, based on a previous model. These pumps are used in the hemofiltration system and to introduce replacement media into the circuit.

These peristaltic pumps (Figure 9) include the Head and Motor supplied by Verderflex with manufacturer's part number *U EZ3R16 100 02*, a motor driver designed specifically for these pumps and the Arduino microcontrollers. In this case, just one Arduino Uno Slave ("b") is enough to control both peristaltic pumps together with the Arduino Uno Master.

Figure 9: Pictures of the peristaltic pumps for the hemofiltration system and the replacement media.

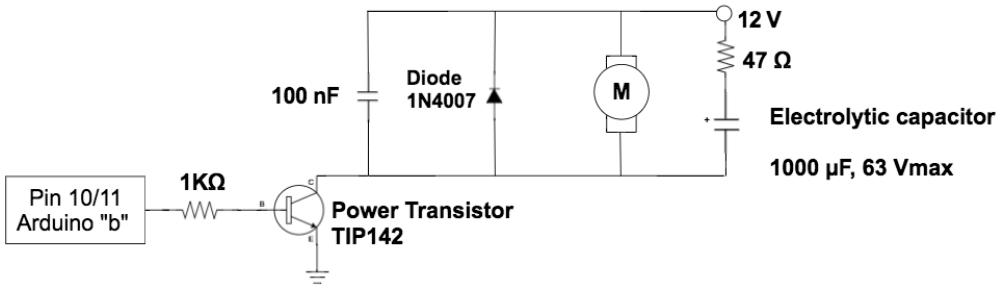

Figure 10: Electronic circuit of the driver from the peristaltic pump. It includes two resistors, a power transistor, two capacitors and a diode.

The motor driver (Figure 10) includes two capacitors to reduce the noise of the signal, a diode (1N4007) to prevent possible inverse voltage from the coils of the motor due to induction and a power transistor (TIP142). This last one is a Darlington transistor PNP or AMP, a semiconductor based on a combination of two bipolar transistors, which behaves as a switch depending on the drop of voltage across it.

For both pumps start and stop buttons are included in the Guided User Interface. Additionally, there is a box where the user can enter a number between 0 and 255 to determine the speed of the pump. If the entered value is not valid an error message will be shown and the pump will stop. A value equal to 0 is equivalent to 0V and 255 means 5V, the maximum voltage, and so maximum speed.

The value is transmitted from Gambas to Arduino Uno Master and then to Arduino Uno Slave "b". The program of the last microcontroller interprets if the speed or any other command (Table 4) is given to pump 1 (RM or replacement media) or 2 (HF or hemofilter). However, in the case of the first peristaltic pump, it is not as simple. It also depends on the measurement from the level sensor. The pump will be active if the level is below the established threshold. However, if the level is enough the pump will immediately become inactive with a speed value of 0 independently of the value entered by the user.

The motor driver is connected to the motor through Faston and Molex connections, to the power supply and the Arduino Uno Slave. In this case, apart from the I^2C connections, pin 10 and 11 of the microcontroller are used. They are characterized by generating a PWM (pulse-width modulation) whose duty cycle depends on the parameter entered by the user to determine the speed of the pumps.

Command	Function
Mb1S/Mb2S	Returns the current state of peristaltic pump 1 or 2
Mb1P/Mb2P	Stops peristaltic pump 1 or 2
Mb1Vxxx/Mb2Vxxx	Speed of peristaltic pump 1 or 2 (0-255 → 0-5V)

Table 4: Commands to control the peristaltic pumps.

Figure 11: Picture of the level sensor used to determine the level of liquid in the reservoir. If there is no enough culture media the peristaltic pump of the replacement media will be activated to inject some extra fluid. The picture shows a recipient (it is not the final reservoir) with the electrodes and the connections to the electric circuit (the circuit is not included).

5.5 Level sensor

The level sensor (Figure 11) is used to determine the amount of media in the reservoir to control the replacement media pump. It is based on the difference in capacitance between the electrodes when liquid is present or absent between them. The electrodes are placed on the outer surface of the reservoir and they do not need to be in direct contact with the inside of the recipient or the liquid.

One of the electrodes is connected to ground and the other one goes to the Simple Schmitt Trigger MM74HC14. In fact, this component contains six oscillators. Two of them are used in chain, connected as shown in Figure 12, to reduce the noise and purify the output square signal and its frequency. That signal will be then connected to Arduino Uno Slave "L" through Pin 5 and its program will decode it to obtain its period and the frequency in MHz.

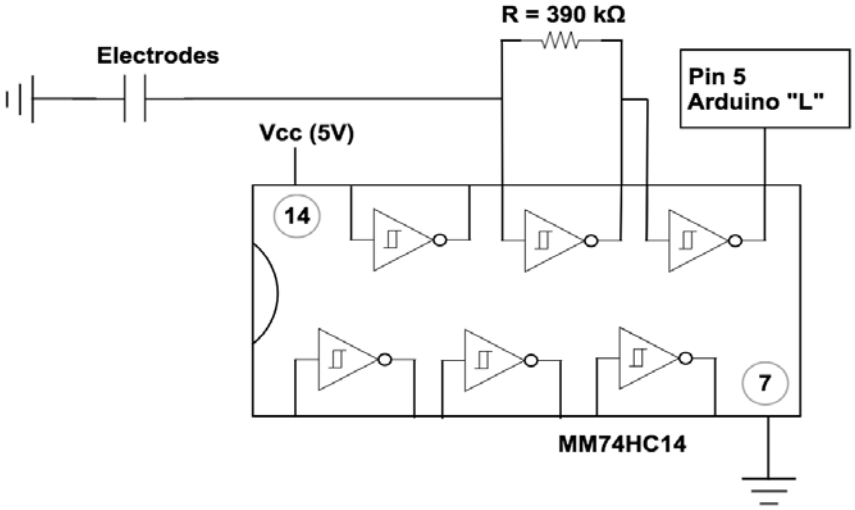

Figure 12: Electronic circuit of the level sensor. The signal from the electrodes is used as the input for the MM74HC14, whose output is a square signal. Its period and frequency depends on the capacitance, which is determined by the level of liquid in the reservoir.

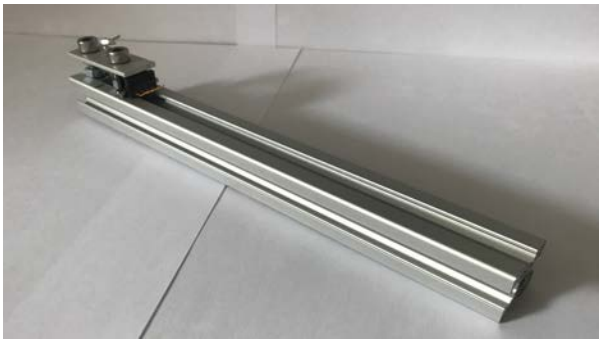

Figure 13: Picture of the weight sensor. It is part of the hemofiltration system used to simulate the kidneys.

Frequency measurement is then sent to Gambas through the Arduino Uno Master and its value is compared with a pre-established threshold, experimentally determined. This comparison determines if the replacement media peristaltic pump is active or inactive. In fact, the Guided User Interface will show a message indicating if the level of the reservoir is enough or if it is not sufficient.

In this case, there is just one command used to control the level sensor: SLR (Table 5). In fact, when the order is sent to the microcontroller, 500 measurements are taken and their average is calculated to reduce possible errors or to eliminate values out of the reference range.

5.6 Weight sensor

Including a weight sensor (Figure 13) in the hemofiltration system has not been done before in a recellularization bioreactor. This novel feature provides automatization to the process. Moreover, as a result, human manipulation is reduced increasing the reproducibility. In this way, the weight of the urine bag can be recorded during the entire process instead of just at the end.

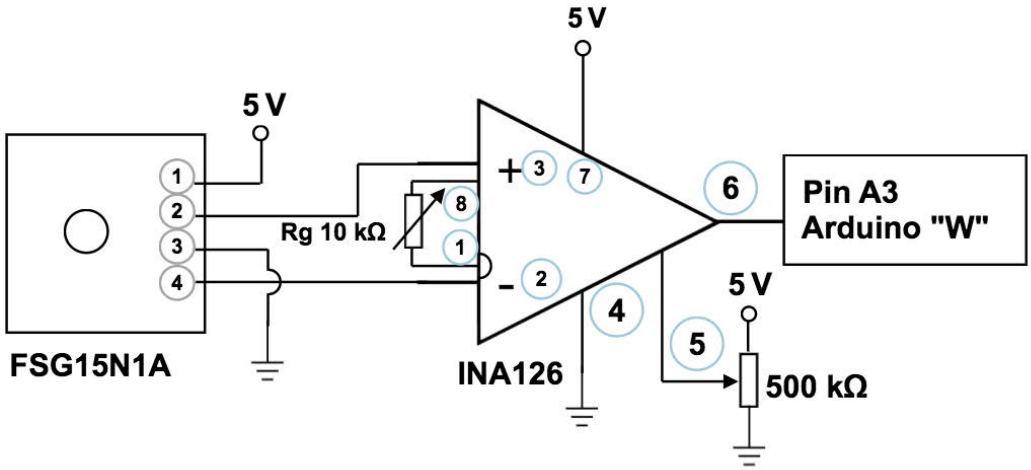

Figure 14: Electronic circuit of the weight sensor. The pressure sensor FSG15N1A measures the weight. Then the signal is transferred to the amplifier (INA126) to finally get to Arduino Uno "W".

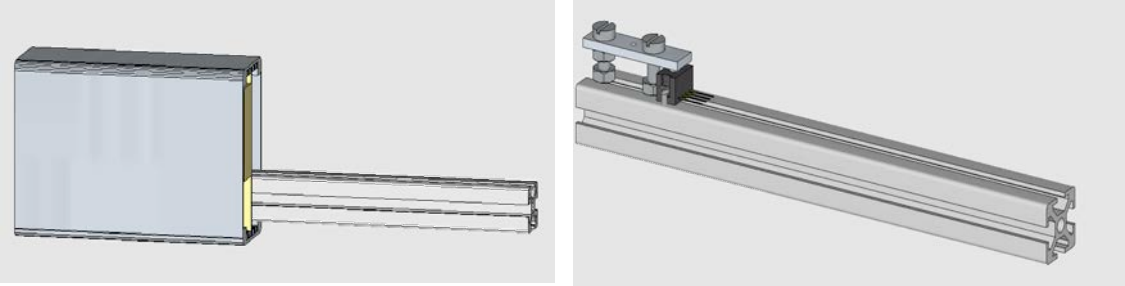

Figure 15: Three-dimensional representation of the weight sensor: (left) inside the protecting box and (right) without the protecting box.

This element is based on the pressure sensor FSG15N1A. It has four different connections: voltage, ground and signal. The two signal outputs are the input for the amplifier INA126. The amplified signal will be read by Pin A3 from Arduino Uno Slave "W". In fact this signal will be a voltage between 0 and 5 volts codified with a number between 0 and 1023. Arduino program transforms that value into a voltage and then into the corresponding weight.

The conversion from voltage to weight requires the use of a calibration curve. It is obtained by loading known weights and measuring the corresponding voltage. Then, a linear regression is applied to obtain the equation of the curve to be applied in the program. Moreover, it is necessary to ensure that the maximum weight to be measured corresponds with 5 volts and no load with 0 volts. To do so the potentiometers of the circuit will be adjusted (Figure 14).

As in the case of the level sensor, there is just one command programmed to control the weight sensor: SWR (Table 5). When Gambas sends this command the level sensor measures the frequency of the signal several times and calculates the average. Then, if Data Acquisition Data is active it will record it in the plaintext file and it will show it on the screen.

Finally, it has to be mentioned that the system that gives rise to the weight sensor is located on a profile bar and will be protected with an external chassis similar to a box (Figure 15).

5.7 pH-meter

The last element that was designed, programmed and built along the second and third phases of the project was a pH-meter. The professional probe and the driver were obtained from Bricogeek. Its precision is of ± 0.1 pH, enough for the purpose of this project.

Figure 16: Picture of the pH-meter. **(A)** The probe. **(B)** The driver. It is used to control the pH at the reservoir. Its measurements determine the injection of acid or basic solution with the corresponding syringe pumps.

There are two different Arduino codes available for this component. On the one hand, the calibration code. It does not require I^2C connection or Gambas. It can be used directly with Arduino. On the other hand, the operating code. When the unique command order pHAR (Table 5) of the sensor is sent, it measures the pH in the reservoir. In fact, it takes ten measurements, sorts them from smaller to larger, removes the two smallest and largest and calculates the average of the other six values.

The pH value is then sent to the Arduino Uno Slave "H" and Arduino Uno Master. Finally Gambas is able to read and decode it. It is later on compared with the threshold pH established by the user in the Guided User Interface. Depending on the comparison acid or basic solution will be added to the reservoir to adjust the pH.

As mentioned above, together with the probe the pH-meter system includes a driver, provided also by Bricogeek. They are joined through a BNC connector. Additionally, the Arduino Uno Slave "H" and the Master are also required in this case. The driver is connected to ground, 5 volts and the Pin A0 from the Arduino Uno Slave "H".

Command	Function
SLR	Returns value measured by the level sensor
SWR	Returns value measured by the weight sensor
pHAR	Returns value measured by the pH-meter

Table 5: Commands to control the level and weight sensors and the pH meter.

6. Socio-economic impact and budget

The budget included in this section corresponds just to phases two and three of the project, the ones developed during this Bachelor Thesis. Then, it can be divided into the different elements that have been designed and built. It has to be considered that only the mechanical and electronic components had costs. Software and design are not reflected in this budget because they did not imply a direct cost.

The cost of the five syringe pumps is displayed in the table below. The final price of the five pumps is 587,56 €. It is important to mention that printing some of the pieces made it possible to obtain the designed model and at the same time decreased the final price. In fact, the cost of a commercial syringe pump with the characteristics required may be between 500 and 1.000€.

Component	Units	Price (€)/Unit	Total price (including taxes)
Threaded shaft	5	2,33	11,65
TB6560 Stepper Motor Driver	5	5,7	28,5
NEMA17 Stepper Motor	5	12,89	67,44
Shaft adapter	5	2,45	17,35
Smooth bars	10	3,65	36,5
Linear bearings	10	0,9	9
625ZZ bearings	5	0,49	3,15
Profile bars	1	14,99	19,99
Arduino Uno (Master and Slaves)	6	15	90
Limit Switch (1 per pump required)	10	1,55	25,97
Nuts	1	4,82	4,82
Set of 5 3D-printed pieces	5	37,31	186,55
Power supply	1	36,64	36,64
Electronics and external box	1	50	50
TOTAL PRICE			**587,56**

Table 6: Budget of the syringe pumps. The total price includes some taxes paid to obtain the components as some of them were bought in other countries.

The next table contains the information relative to the peristaltic pumps. Compared with the pumps of Masterflex (3.070€) used in the first phase of the project they are considerably less expensive.

Component	Units	Price (€)/Unit	Total price (including taxes)
Head + Motor (Verderflex)	2	295,6	591,2
Arduino Uno (Slave)	1	15	15
Electronics and box	1	50	50
TOTAL PRICE			**656,2**

Table 7: Budget of the peristaltic pumps. There is just one Arduino Uno included in the budget as it was enough to control both pumps and the master is included on Table 6.

The budget of the level sensor, the weight sensor and the pH-meter is included in the following tables (Tables 8-10).

Component	Units	Price (€)/Unit	Total price (including taxes)
Pack 100 electrodes	1	56,65	56,65
Arduino Uno (Slave)	1	21	21
Electronics and box	1	50	50
TOTAL PRICE			**127,65**

Table 8: Budget of the level sensor. There is just one Arduino Uno included in the budget as the master is included on Table 6. Just two electrodes are required in the setup.

Component	Units	Price (€)/Unit	Total price (including taxes)
Amplifier INA126	1	2,85	2,85
FSG15N15	1	80,42	80,42
Arduino Uno (Slave)	1	21	21
Electronics and box	1	50	50
TOTAL PRICE			**154,27**

Table 9: Budget of the weight sensor. There is just one Arduino Uno included in the budget as the master is included on Table 6.

Component	Units	Price (€)/Unit	Total price (including taxes)
Probe + Driver	1	59,9	79,68
Arduino Uno (Slave)	1	15	15
Electronics and box	1	50	50
TOTAL PRICE			**144,68**

Table 10: Budget of pH-meter. There is just one Arduino Uno included in the budget as the master is included on Table 6.

Finally, Table 11 groups the cost of all elements. At the end, 1.670,56€ were invested during phases two and three.

Component	Total price
Syringe Pumps x5	587,76
Peristaltic Pumps x2	656,2
Level Sensor	127,65
Weight Sensor	154,27
pH-meter	144,68
TOTAL PRICE	**1.670,56**

Table 11: Total budget of phases two and three from the project: five syringe pumps, two peristaltic pumps, a pH-meter, a level sensor and a weight sensor.

The calculations above show that phases two and three were not highly expensive. It has to be considered that the real investment will be required in the following phases of the project, when biological experiments will be performed and particularly in the last one. For example, recellularization of a liver implies, first, to harvest and decellularized it. In the first case a surgery room, qualified personnel and surgical instruments will be required, increasing the cost of each perfusion.

On the other hand, there are different elements of the bioreactor, such as the syringe and the peristaltic pumps that do not have to be bought for each perfusion. Their sterilization prior to use will allow to utilize them several times. Thus, the initial investment is repaid if numerous perfusions are performed. For that to be possible, the materials chosen must be durable as well as biocompatible.

Even when the viability of the project will have to be evaluated once it is finished, the cost is not the most important point. A recellularization bioreactor will be used to obtain bioengineered organs to save human lives. Organ shortage is an increasing problem in today's society, especially due to the aging of population. Furthermore, traditional organ transplantation also implies a high cost. Surgery, immunosuppressing therapies, analysis to check compatibility and other expensive procedures are required, without a guarantee of success.

It is also important to consider that this project is just the initial starting point. Further investigation will be also required until functional livers can be transplanted into humans. Then, once the recellularization bioreactor is obtained, its price may decrease. Bioreactors, as any other technological advance, need time to become low-cost. Advances in other fields as materials science and engineering, cell culture or stem cell culture may also reduce the final price of whole-organ bioengineering.

Once functional bioengineered livers, or other organs, can be obtained life expectancy and quality of humans will increase being possible to cure several diseases. Additionally, cultural and ethical considerations that limit organ donation will not have to be considered in this case as the organ scaffolds can be obtained from an animal source. Modern society considers that humans are the most valuable living beings on Earth. Then, using animals for experimentation and to obtain the required scaffolds, both under a legal framework, may be justified as it ultimately saves lives.

The analysis of the socio-economic impact of this project has different arguments that demonstrate that although it may be expensive it is a necessity in today's society. As a conclusion, when health is involved, the conventional analysis between costs and benefits are not adequate. The inherent value of recellularization bioreactors is much higher than its cost.

7. Legislation and GMP

The EMA[36] (European Medicines Agency) and the AEMPS[37] (Agencia Española de Medicamentos y Productos Sanitarios) are the agencies in charge of the regulation of medical treatments and products in the European Union and Spain, respectively.

A medicinal product is defined as any substance or any combination of different substances that are administered with the purpose of making a diagnosis, to treat or to prevent a human disease. If they are based on gene therapy, somatic therapy or tissue engineering they are called Advanced Therapy Medicinal Products (ATPMs). Then, bioengineered organs are considered to be ATMPs. Furthermore, they involve engineered tissues used to replace a diseased human organ with the use of cells subjected to substantial manipulation.

These products have a specific regulation for their authorization, supervision and pharmacovigilance. Marketing and clinical studies need to be authorized and their quality, safety and efficacy need to be demonstrated. At the same time, GMP (manufacturing and quality control), GLP (pre-clinical development) or and GCP (clinical trials) need to be applied. On the one hand, GLP or Good Laboratory Practices are required to ensure and protect the integrity and truthfulness of scientific data in non-clinical health studies. On the other hand, GCP or Good Clinical Practices are needed in clinical trials in which humans are involved as it defines the ethical and scientific standards.

Finally, GMP or Good Manufacturing Practices are the policies applied to the manufacturing and quality control processes. They ensure that medical products are homogeneously fabricated and under the appropriate control according to the legislation. In this way, the risk associated with the production process, such as contamination, is reduced. In the case of recellularization process, in order to guarantee GMP, automation and reproducibility, bioreactors must be unquestionably used.

Additionally, if the products are derived from biotechnology, as it is the case of bioengineered organs, they must undergo a process to be centrally authorized. In this case, the CHMP (Committee for Medical Products for Human use) as part of the EMA must evaluate the product, as well as the documentation that the laboratory has to send, and decide if its commercialization can be authorized or not. This process is known as the Centralized Procedure and implies the approval of the product in all European countries. It has an approximate duration of 210 days and it is based on an evaluation of benefits and risks of the product under study.

The Centralized Procedure starts with the application and the study of the required documentation. In case needed, the manufacturer's methods to ensure control will be evaluated. Then, CAT (Committee for Advanced Therapies) renders a draft decision and CHMP finally resolves if the product is authorized or not. There will also be a period of time after the resolution in which the manufacturer can ask for a new examination if the authorization has not been favorable.

8. Future perspectives and conclusion

Recellularization process could be performed just using the tubing, reservoir, bioreactor vessel, Masterflex peristaltic pumps, pressure and flow sensors. However, the outcome will considerably improve if more sophisticated control of all parameters is achieved and the microenvironment is mimicked better.

As part of this Bachelor Thesis, each of the components was individually tested. Those elements whose functioning is interconnected were also analyzed together. It is the case of the pH-meter and two syringe pumps as well as the level sensor and the replacement media peristaltic pump. The performance of the system was successfully verified.

As the project is already organized in phases, the next step will be to develop the fourth phase. The different components have been designed, programmed and built at the *Hospital Universitario Gregorio Marañón*. However, recellularization of liver scaffolds will take place at the *Instituto de Investigación Sanitaria de Aragón*. Then, the different elements will have to be transported to continue with the project.

There are other important considerations to bear in mind. The fifth phase of the project will eliminate the incubator as temperature, pH and oxygenation method. The new one will only inject oxygen in the line corresponding to the hepatic artery. Then, two pH-meters will be required instead of just one. The first one will be placed in the reservoir, as it is now, and the other will be located after the oxygenation system. Furthermore, the probe of the pH-meter will have to be changed in approximately one year, as indicated by the supplier, and all the sensors should the calibrated again after some perfusions.

The objectives of this Bachelor Thesis have been achieved. The pumps and sensors, as indicated above, have been designed, programmed and built. Moreover, all the information related with the second and third phases of the project has been well documented. In fact, most of the relevant data is contained in this document. On the other hand, it is also important to mention that the last phase of the project, metabolic monitoring, has been already started. It will be performed in parallel with the other phases and in collaboration with the center for biomedical technology (CTB).

To sum up, recellularization is a complex process but it will be useful to obtain new drug discovery platforms and bioengineered organs for transplantation. In that way both applications will improve human health allowing scientists to find new treatments and cures for different diseases. In my view, whole-organ bioengineering is currently the only real alternative to significantly increase the pool of organs for transplantation. Although a lot of research and investigation is still required to obtain functional organs, it is a promising technique that will ultimately save lives.

In conclusion, bioreactors play a major role to achieve the goal of Tissue Engineering and Regenerative Medicine and will be the key to solve organ transplantation problems.

9. Annexes: Arduino codes

There are two different types of Arduino codes: master and slave. They are based on previous codes developed by Dr. Juan Francisco del Cañizo and adjusted to the requirements of this project.

There are two different directions in which information flows, both based on interrupts. On the one hand, the master may send information through the corresponding bus direction (DirBus) as a string (Texto), which comes from the serial as StrS. This is performed with the function "trasmite" on the master. Then, in the code of the corresponding slave the function "receive", registered as an event in the setup, is executed to decode the command sent and perform its function.

On the other hand, the slave may send information back to the master as requested. In this case the first function executed is "recibe" at the master, determining the bus direction (DirBus) and the number of bytes that will be received from the slave (numchars). Then, the "request" event, also registered at the setup, is executed to obtain the required information (Rstr and cadena) to be sent back to the master.

9.1 Master

```
// Arduino code to control the Arduino Uno Master

#include <Wire.h>       // This line includes the library called Wire, which allows the communication with I2C

String St="";           // String variable required in the function recibe (information from slave to master)
char cadena[20];        // Char variable required in the function trasmite (information from master to slave)
char ByteR;             // Byte read by the serial
String StrS = "";       // String to store incoming data
boolean SFin = false;   // Variable used to determine if the string is complete or not

// ************************* SETUP ******************************

void setup() {

  Wire.begin();         // Join I2C bus with address Wdir (optional in the master)
  Serial.begin(9600);   // Start serial for information flow.
  St.reserve(100);      // Reserve function allows the manipulation of the string St with a size of 100 bytes

}

// ************************* END SETUP ******************************

// ************************* LOOP ******************************

void loop() {

  if (Serial.available() > 0)

  // It reads the incoming byte
  // If the byte corresponds with '\n' it stops and SFin becomes true
    {

      ByteR = (char)Serial.read();
```

```
    StrS += ByteR;
    if (ByteR == '\n') SFin = true;
   }

if (SFin == true) {
// If the chain read has finished it starts decoding
// It is based on a switch and the different cases depend on the second letter of the command

    switch(StrS[1])
      {

       case 'A':                    // Syringe pump A, bus direction 10
          trasmite(10, StrS);       // Transmission of the command StrS coming from the serial
          delay(100);               // Delay of 100 milliseconds
          recibe(10,30);            // Reads the information sent back from the slave with a length of 30 characters
       break;

       case 'B':                    // Syringe pump B, bus direction 11
          trasmite(11, StrS);
          delay(100);
          recibe(11,30);
       break;

       case 'C':                    // Syringe pump C, bus direction 12
          trasmite(12, StrS);
          delay(100);
          recibe(12,30);
       break;

       case 'D':                    // Syringe pump D, bus direction 13
          trasmite(13, StrS);
          delay(100);
          recibe(13,30);
       break;

       case 'E':                     // Syringe pump E, bus direction 14
          trasmite(14, StrS);
          delay(100);
          recibe(14,30);
       break;

       case 'b':                    // Peristaltic pumps (b1 and b2), bus direction 15
          trasmite(15, StrS);
          delay(100);
          recibe(15,30);
       break;

       case 'H':                    // pH meter (A), bus direction 16
          trasmite(16, StrS);
          delay(100);
          recibe(16,30);
       break;

       case 'L':                    // Level sensor, bus direction 17
          trasmite(17, StrS);
          delay(100);
          recibe(17,30);
       break;

       case 'W':                    // Weight sensor, bus direction 18
          trasmite(18, StrS);
          delay(100);
```

```
          recibe(18,30);
          break;
         }

      // SFin becomes false so that the switch loop is stopped and the code just reads the incoming bytes for a new
      // command

       StrS="";
       SFin = false;
      }

}

// ************************* END LOOP ******************************

// ************************* SUB TRASMITE ******************************

void trasmite(int DirBus, String Texto)  {
// Function required to send information (Texto) from the master to the slave through DirBus direction bus

            Wire.beginTransmission(DirBus);                  // Start transmitting
            Texto.toCharArray(cadena, Texto.length() + 1);   // The string Texto is copied into the char cadena
            Wire.write(cadena);                              // Writes the data in cadena
            Wire.endTransmission();                          // Stop transmitting

}

// ************************* END TRASMITE ******************************

// ************************* SUB RECIBE ******************************

void recibe(int DirBus, int numchars){
// Function required to receive information from the slave through DirBus direction bus

          Wire.requestFrom(DirBus, numchars);
          // numchars determines the number of bytes read by the master and sent by the slave

          St = "";

          while (Wire.available())
            {
              char c = Wire.read();    // Receive a byte as character
              St = St + c;             // Store bytes at St string
            }

          // The slave may send less bytes than numchars value, in that case St is filled with extra bytes

          Serial.println(St);

}

// ************************* END RECIBE ******************************
```

9.2 Syringe pumps

```
// Arduino code to control the syringe pumps. This code is for the syringe pump A.
// To adjust it to B, C, D or E, the bus direction (Wdir) and variable 'disp' will be changed to the corresponding
// one.
```

```
#define SEN 2            // Pin to enable the motor (high = off / low = on)
#define SSTEP 3          // Pin to determine the steps to be moved (high = movement / low =  no movement)
#define SDIR 4           // Pin to determine the direction of the movement (high = backward / low = forward)
#define FIN2 12                  // Pin for the limit switch (FIN2) (high = not activated / low = activated)
#define MAXPASOS 27000   // Maximum number of steps that the syringe can be moved

#include <Wire.h>      // This line includes the library called Wire, which allows the communication with I2C
int Wdir = 10;         // Direction of the peristaltic pump in the I2C bus
char Disp = 'A';       // Instead of using the letter assigned to each pump, a variable is defined making it easier to
                       // adjust the code for other pumps

char ByteR;            // Byte read by the serial line to be stored in StrS

String Rstr = "";      // String to store output information
char cadena[21];       // Variable to store the output information in the requestEvent

String StrS = "";      // String to store input information
String comando;        // String to store the information sent by the master in the receiveEvent
String comando2;       // String to store the information sent by the master in the receiveEvent

unsigned long tpaso;         // Time at a specific moment
unsigned long contpasos;   // Steps moved
unsigned long numpasos;   // Steps to be moved
unsigned long tant;           // Initial time of a period
boolean valpos = false;      // Variable to determine if the position is valid, if the zero position has been recognized
long periodo = 2000;         // Variable to determine the period.
                             //Establishes 2000 microseconds as the default time between pulses
long pos;                    // Position of the syringe in steps

// ************************** SETUP ********************************

void setup(){

  Wire.begin(Wdir);                  // Join I2C bus with address Wdir
  Wire.onRequest(requestEvent);    // It registers the function "requestEvent" that will be called by the master to
                                     // obtain information from the slave
  Wire.onReceive(receiveEvent);    // It registers the function "receiveEvent" that will be called when the slave
                                    // receives information
  Serial.begin(9600);                // Start the serial for the output information

  StrS.reserve(50);              // Reserve function allows the manipulation of the string StrS with a size of 50 bytes
  Rstr.reserve(100);             // Reserve function allows the manipulation of the string Rstr with a size of 100 bytes

  pinMode(SDIR, OUTPUT);          // Defines the direction pin as output
  pinMode(SSTEP, OUTPUT);       // Defines the steps pin as output
  pinMode(SEN, OUTPUT);          // Defines the enabling pin as output
  pinMode(FIN2, INPUT);           // Defines the limit switch pin as input

  digitalWrite(SDIR, HIGH);       // Establishes backward direction
  digitalWrite(SEN, HIGH);        // Stops the stepper motor

  numpasos = 100;          // The number of steps to be moved has a minimum value of 100
  pos = MAXPASOS;     // Initial the position is at the maximum number of steps so that all the steps can be
                           // moved
  tant = micros();          // Function "micros()" returns the time since the execution of program began in
                           // microseconds

}

// ************************** END SETUP ********************************

// ************************** LOOP ********************************
```

```
void loop() {

tpaso = micros() - tant;
if((tpaso >= periodo) && (digitalRead(SEN)== LOW))  // The period has finished and the motor is enabled
   {

    if(digitalRead(SDIR) == 1 && digitalRead(FIN2)) // Backward direction and has not reached the limit switch
      {
        digitalWrite(SSTEP, HIGH); // Moves a step backward and counts a increases a position
        contpasos++;
        pos++;
      }

    if(digitalRead(SDIR) == 0 && pos > 0 ) // Forward direction and has not reached position 0
      {
       digitalWrite(SSTEP, HIGH); // Moves a step forward and counts a decreases a position
       contpasos++;
       pos--;
      }

    tpaso = 0;
    tant = micros();
   }

   else

   // Until here, if there are steps left to be moved and the limit switch has been activated, the direction is changed
   // to complete the steps. The next lines are required to avoid it:

   {
    if(digitalRead(FIN2)== 0 && digitalRead(SDIR) == 1) // Limit switch activated and backward direction
      {
       digitalWrite(SDIR,LOW);     // Forward direction activated
       numpasos = 0;                // No more steps moved
       contpasos = 0;              // Cero steps moved
       pos = MAXPASOS;             // The maximum number of steps has been reached
       valpos = true;              // it is a valid position
      }

    if(pos == 0 && digitalRead(SDIR) == 0) // There are no more steps to be moved and there is forward direction
      {
       digitalWrite(SDIR,HIGH);     // Backward direction activated
       numpasos = 0;                // No more steps moved
       contpasos = 0;               // Cero steps moved
      }
    digitalWrite(SSTEP, LOW);      // No more steps to be moved
   }

   if (contpasos >= numpasos)       // Steps to be moved or more have been already performed
    {
      digitalWrite(SEN, HIGH);     // The stepper motor is disabled
      contpasos = 0;
    }
}

// ************************** END LOOP ********************************

// ************************** SUB REQUEST_EVENT ********************************

// It transforms the Rstr string into char to be written in the I2C bus that will send information from the slave to the
// master
```

```
void requestEvent() {

  Rstr.toCharArray(cadena, Rstr.length() + 1);   // The string Rstr is copied into the char cadena
  Wire.write(cadena);                            // Writes the data in cadena
  Serial.println(cadena);
  Rstr = "";
}

// ************************** END REQUEST_EVENT **********************************

// ************************* SUB RECEIVE_EVENT ***************************

// It receives the information sent by the master

void receiveEvent(int howMany) {   // howMany defines the number of bytes read by the slave

  while (Wire.available())
  // While information is being sent the loop is performed
  // Stores the character read in c and adds it to the string comando

    {
     char c = Wire.read();
     comando = comando + c;
    }

  comando2 = comando.substring(0,4);   // comando2 stores characters from 0 (inclusive) to 4 (exclusive)
  StrS = comando;                      // comando is stored as StrS and leeStr is called
  leeStr();
  comando = "";

}

// ************************** END RECEIVE_EVENT **********************************

// ************************* SUB LEESTR ******************************

void leeStr(){

 long a, b, c, d, e, f, g, i;   // Auxiliary variables to decode the number of steps to be moved

      if ( StrS[0]== 'C' && StrS[1]== Disp) {  // A switch is used to perform the different commands

        switch(StrS[2]){

          case 'M': // CAMXXXXXX moves the stepper motor XXXXXX steps
          // The digits are transformed from ASCII code into the equivalent decimal character
          // They are multiplied to give the variable the correct value and all of them are added:

            a=(long(StrS[3])-48)*100000;
            b=(long(StrS[4])-48)*10000;
            c=(long(StrS[5])-48)*1000;
            d=(long(StrS[6])-48)*100;
            e=(long(StrS[7])-48)*10;
            f=(long(StrS[8])-48);
            g= a + b + c + d + e + f;
            numpasos = g;                    // Number of steps to be moved (loop)
           digitalWrite(SEN, LOW);           // Enables the stepper motor

            Rstr = "&M*" + String(numpasos, DEC) + "*";
            Serial.print ("&"); Serial.print (Disp);
            Serial.print ("M*"); Serial.println (numpasos);
          break;
```

```
case 'D': // Establishes the direction of the movement
  switch(StrS[3]){

    case '+':       // Forward direction = CAD+
      digitalWrite(SDIR, LOW);
      Rstr = "&D+*" + String(digitalRead(SDIR),DEC) + "*";

      Serial.print ("&"); Serial.print (Disp);
      Serial.print ("D+*"); Serial.println(digitalRead(SDIR));
    break;

    case '-':       // Backward direction = CAD-
      digitalWrite(SDIR, HIGH);
      Rstr = "&D-*" + String(digitalRead(SDIR),DEC) + "*";

      Serial.print ("&"); Serial.print (Disp);
      Serial.print ("D-*"); Serial.println(digitalRead(SDIR));
   break;
  }

break;

case 'V': // CAVXXXXXX determines a period of XXXXXX microseconds between pulses
// The digits are transformed from ASCII code into the equivalent decimal character
// They are multiplied to give the variable the correct value and all of them are added:

  a=(long(StrS[3])-48)*100000;
  b=(long(StrS[4])-48)*10000;
  c=(long(StrS[5])-48)*1000;
  d=(long(StrS[6])-48)*100;
  e=(long(StrS[7])-48)*10;
  f=(long(StrS[8])-48);
  g= a + b + c + d + e + f;
  periodo = g;

  if(periodo < 200) periodo = 200; // The minimum period is 200

  Rstr = "&V*" + String(periodo, DEC) + "*";

  Serial.print ("&"); Serial.print (Disp);
  Serial.print ("V*"); Serial.println(periodo);
break;

case 'S':  // Returns the current step of the syringe pump
    Rstr = "&S*" + String(digitalRead(FIN2),DEC);
    Rstr = Rstr + "*" + String(digitalRead(SDIR),DEC);
    Rstr = Rstr + "*" + String(periodo, DEC);
    Rstr = Rstr + "*" + String(pos, DEC);
    Rstr = Rstr + "*" + String(valpos, DEC) + "*";

    Serial.print ("&C"); Serial.print (Disp);
    Serial.print ("S*FAT="); Serial.print(digitalRead(FIN2));
    Serial.print ("*DIR="); Serial.print(digitalRead(SDIR));
    Serial.print ("*PER="); Serial.print(periodo);
    Serial.print ("*Pos="); Serial.print(pos);
    Serial.print ("*PVa="); Serial.println(valpos);
break;

case 'P':  // Disables the stepper motor
  digitalWrite(SEN, HIGH);
  Rstr = "&P*" + String(digitalRead(SEN),DEC) + "*";
```

```
          Serial.print ("&"); Serial.print (Disp);
          Serial.print("P*"); Serial.println(digitalRead(SEN));
         break;

         case 'G':  // Enables the stepper motor
          digitalWrite(SEN, LOW);
          Rstr = "&G*" + String(digitalRead(SEN),DEC) + "*";

          Serial.print ("&"); Serial.print (Disp);
          Serial.print("G*"); Serial.println(digitalRead(SEN));
         break;
        }
      }
}

// ************************* END LEESTR ******************************
```

9.3 Peristaltic pumps

```
// Arduino code to control both peristaltic pumps.

#include <Wire.h>      // This line includes the library called Wire, which allows the communication with I2C
int Wdir = 15;         // Direction of the pH meter in the I2C bus

char ByteR;            // Byte read by the serial line to be stored in StrS

String Rstr = "";      // String to store output information
char cadena[21];       // Variable to store the output information in the requestEvent

String StrS = "";      // String to store input information
String comando;        // String to store the information sent by the master in the receiveEvent
String comando2;       // String to store the information sent by the master in the receiveEvent

int valPC1 = 0;        // Value provided by the user for pump 1 (0-255)
int valPC2 = 0;        // Value provided by the user for pump 2 (0-255)

const int D1out = 10;     // Output pin for pump 1
const int D2out = 11;     // Output pin for pump 2

// ************************* SETUP ******************************

void setup(){

  Wire.begin(Wdir);                  // Join I2C bus with address Wdir
  Wire.onRequest(requestEvent);      // It registers the function "requestEvent" that will be called by the master to
                                     // obtain information from the slave
  Wire.onReceive(receiveEvent);      // It registers the function "receiveEvent" that will be called when the slave
                                     // receives information
  Serial.begin(9600);                // Start the serial for the output information

  StrS.reserve(50);          // Reserve function allows the manipulation of the string StrS with a size of 50 bytes
  Rstr.reserve(100);         // Reserve function allows the manipulation of the string Rstr with a size of 100 bytes



  pinMode(D1out, OUTPUT);       // Pin 10 initialized as output for pump 1
  pinMode(D2out, OUTPUT);       // Pin 11 initialized as output for pump 2
```

```
  analogWrite(D1out, 0);        // Pump 1 initially stopped
  analogWrite(D2out, 0);        // Pump 2 initially stopped

}

// ************************** END SETUP ******************************

// ************************** LOOP ******************************

// Void loop stores in StrS string the characters read from the serial:

void loop() {

   if (Serial.available() > 0)
   // Serial.available provides the number of characters available to be read

     {
      ByteR = (char)Serial.read();  // "Serial.read" reads the data available and char transforms it into char data type
      StrS += ByteR;
     }

}

// ************************** END LOOP ******************************

// ************************** SUB REQUEST_EVENT ******************************

// It transforms the Rstr string into char to be written in the I2C bus that will send information from the slave to the
// master

void requestEvent() {

  Rstr.toCharArray(cadena, Rstr.length() + 1);      // The string Rstr is copied into the char cadena
  Wire.write(cadena);                               // Writes the data in cadena
  Serial.println(cadena);
  Rstr = "";

}

// ************************** END REQUEST_EVENT ******************************

// ************************** SUB RECEIVE_EVENT ******************************

// It receives the information sent by the master

void receiveEvent(int howMany) {   // howMany defines the number of bytes read by the slave

  while (Wire.available())
  // While information is being sent the loop is performed
  // Stores the character read in c and adds it to the string comando

    {
     char c = Wire.read();
     comando = comando + c;
    }

  comando2 = comando.substring(0,4);    // comando2 stores characters from 0 (inclusive) to 4 (exclusive)
  StrS = comando;                       // comando is stored as StrS and leeStr is called

  leeStr();
  comando = "";
}
```

```
// ************************** END RECEIVE_EVENT ************************************

// ************************** SUB LEESTR *******************************

void leeStr(){
  int a, b, c, d;   // Auxiliary variables to obtain the speed of the pump from the value entered by the user (GUI)
  if ( StrS[0]== 'M' && StrS[1]== 'b') {

          if(StrS[2]== '1'){

            if(StrS[3]=='P'){  // Stops peristaltic pump 1 (b1)
              valPC1 = 0;
              analogWrite(D1out, valPC1);
              Serial.print("&Mb1P*"); Serial.print(valPC1); Serial.println("*");
              Rstr = "&Mb1P*" + String(valPC1, DEC) + "*";
              }

            if(StrS[3]=='V'){
            // The digits are transformed from ASCII code into the equivalent decimal character
            // They are multiplied to give the variable the correct value and all of them are added:

               a=(int(StrS[4])-48)*100; b=(int(StrS[5])-48)*10; c=(int(StrS[6])-48);
               d=a+b+c;

               if(d>255) d=255;     // Maximum speed corresponds with a value of 255
               if(d<0) d=0;         // Negative values are not valid
               valPC1 = d;

               analogWrite(D1out, valPC1);
               Serial.print("&Mb1V*"); Serial.print(valPC1); Serial.println("*");
               Rstr = "&Mb1V*" + String(valPC1, DEC) + "*";
             }

           if(StrS[3]=='S'){        // Gives back the state of peristaltic pump 1 (b1)
              Serial.print("&Mb1S*"); Serial.print(valPC1); Serial.println("*");
              Rstr = "&Mb1S*" + String(valPC1, DEC) + "*";
             }
           }

          if(StrS[2]== '2'){

            if(StrS[3]=='P'){       // Stops peristaltic pump 2 (b2)
              valPC2 = 0;
              analogWrite(D2out, valPC2);
              Serial.print("&Mb2P*"); Serial.print(valPC2); Serial.println("*");
              Rstr = "&Mb2P*" + String(valPC2, DEC) + "*";
              }

            if(StrS[3]=='V'){
            // The digits are transformed from ASCII code into the equivalent decimal character
            // They are multiplied to give the variable the correct value and all of them are added:

               a=(int(StrS[4])-48)*100; b=(int(StrS[5])-48)*10; c=(int(StrS[6])-48);
               d=a+b+c;

               if(d>255) d=255;     // Maximum speed corresponds with a value of 255
               if(d<0) d=0;         // Negative values are not valid
               valPC2 = d;

               analogWrite(D2out, valPC2);
               Serial.print("&Mb2V*"); Serial.print(valPC2); Serial.println("*");
               Rstr = "&Mb2V*" + String(valPC2, DEC) + "*";
```

```
            }

           if(StrS[3]=='S'){  // Gives back the state of peristaltic pump 2 (b2)
              Serial.print("&Mb2S*"); Serial.print(valPC2); Serial.println("*");
              Rstr = "&Mb2S*" + String(valPC2, DEC) + "*";
             }
           }
        }

}

// ************************* END LEESTR ******************************
```

9.4 Level sensor

```
// Arduino code to control the level sensor.

// There is a unique command SLR that returns "&SLR*Mhz*" as a string to be read by Gambas.

#include <Wire.h>    // This line includes the library called Wire, which allows the communication with I2C
int Wdir = 17;       // Direction of the level sensor in the I2C bus

char ByteR;          // Byte read by the serial line to be stored in StrS

String Rstr = "";    // String to store output information
char RstrP1[4];      // Auxiliary variable to store output information
char RstrP2[5];      // Auxiliary variable to store output information
char RstrP3[6];      // Auxiliary variable to store output information
char RstrP4[7];      // Auxiliary variable to store output information
char cadena[21];     // Variable to store the output information in the requestEvent

String StrS = "";    // String to store input information
String comando;      // String to store the information sent by the master in the receiveEvent
String comando2;     // String to store the information sent by the master in the receiveEvent


unsigned long durationA;  // Auxiliary variable to calculate the period
unsigned long durationB;  // Auxiliary variable to calculate the period
unsigned long periodo;    // Variable to store the average period measured
float sumaper;            // Auxiliary variable to calculate the averaged period

int pin = 5;              // Arduino Pin 5 defined for the input signal
float Mhz;                // Stored frequency value
int cont;

// ************************* SETUP ******************************

void setup() {

  Wire.begin(Wdir);                  // Join I2C bus with address Wdir
  Wire.onRequest(requestEvent);      // It registers the function "requestEvent" that will be called by the master to
                                     // obtain information from the slave
  Wire.onReceive(receiveEvent);      // It registers the function "receiveEvent" that will be called when the slave
                                     // receives information

  Serial.begin(9600);                // Start the serial for the output information
  StrS.reserve(50);                  // Reserve function allows the manipulation of the string StrS with a size of 50 bytes
  Rstr.reserve(100);                 // Reserve function allows the manipulation of the string Rstr with a size of 100 bytes
```

```
}

// ************************* END SETUP ******************************

// ************************* LOOP ******************************

// Void loop stores in StrS string the characters read from the serial:

void loop() {

   if (Serial.available() > 0)
   // Serial.available provides the number of characters available to be read

     {
      ByteR = (char)Serial.read();    // "Serial.read" reads the data available and char transforms it into char data type
      StrS += ByteR;
     }

 }

// ************************* END LOOP ******************************

// ************************* SUB REQUEST_EVENT ******************************

// It transforms the Rstr string into char to be written in the I2C bus that will send information from the slave to the
// master

void requestEvent() {

  Rstr.toCharArray(cadena, Rstr.length() + 1);       // The string Rstr is copied into the char cadena
  Wire.write(cadena);                                // Writes the data in cadena
  Serial.println(cadena);
  Rstr = "";

}

// ************************* END REQUEST_EVENT ******************************

// ************************* SUB RECEIVE_EVENT ******************************

// It receives de information sent by the master

void receiveEvent(int howMany) {   // howMany defines the number of bytes read by the slave

  while (Wire.available())
  // While information is being sent the loop is performed
  // Stores the character read in c and adds it to the string comando

    {
     char c = Wire.read();
     comando = comando + c;
    }

  comando2 = comando.substring(0,4);    // comando2 stores characters from 0 (inclusive) to 4 (exclusive)
  StrS = comando;                       // comando is stored as StrS and leeStr is called
  leeStr();
  comando = "";

}

// ************************* END RECEIVE_EVENT ******************************
```

```
// ************************* SUB LEESTR *******************************

void leeStr(){


 if ( StrS[0]== 'S' && StrS[1]== 'L' && StrS[2]=='R') {  // If the command sent corresponds with "SLR":

     for (cont = 0; cont<500; cont++)
     // It gets 500 samples measured by the sensor
     // Calculates the period as the addition of time that the pulse is high and low (pulseIn)
     // Stores all the measurements in sumaper

       {
        durationA = pulseIn(pin, HIGH); //
        durationB = pulseIn(pin, LOW);
        periodo = durationA + durationB;
        sumaper = sumaper + periodo; //
        cont = cont;
       }

     sumaper = sumaper / 500;   // It performs the media of the 500 measurements stored at sumaper
     Mhz = 1000 / sumaper;      // Calculates the frequency in Mhz from the averaged period (sumaper)
     sumaper = 0;               // Variables are reinitialized for the next measurement
     cont = 0;

     // As Mhz is a float variable it has to be converted into a char (dtostrf) and then into a string
     // There are four possibilities in this case, as the number of digits of the variable increases if the
     // value is equal or higher than 10, 100 and 1000, respectively:

     if (Mhz < 10 )
       {
        dtostrf(Mhz, 4, 2, RstrP1);
        Rstr = "&SLR*" + String(RstrP1) + "*";
       }

     if (Mhz >= 10 && Mhz <100)
       {
        dtostrf(Mhz, 5, 2, RstrP2);
        Rstr = "&SLR*" + String(RstrP2) + "*";
       }

     if (Mhz >= 100 && Mhz <1000)
       {
        dtostrf(Mhz, 6, 2, RstrP3);
        Rstr = "&SLR*" + String(RstrP3) + "*";
       }

     if (Mhz >= 1000 && Mhz <10000)
       {
        dtostrf(Mhz, 7, 2, RstrP4);
        Rstr = "&SLR*" + String(RstrP4) + "*";
       }
 }

 }

// ************************* END LEESTR *******************************
```

9.5 Weight sensor

```
// Arduino code to control the level sensor.

//There is a unique command SWR that returns "&SWR*peso*" as a string to be read by Gambas.

#include <Wire.h>       // This line includes the library called Wire, which allows the communication with I2C
int Wdir = 18;          // Direction of the weight sensor in the I2C bus

char ByteR;             // Byte read by the serial line to be stored in StrS

String Rstr = "";       // String to store output information
char RstrP1[5];         // Auxiliary variable to store output information
char RstrP2[6];         // Auxiliary variable to store output information
char RstrP3[7];         // Auxiliary variable to store output information
char RstrP4[8];         // Auxiliary variable to store output information
char cadena[21];        // Variable to store the output information in the requestEvent

String StrS = "";       // String to store input information
String comando;         // String to store the information sent by the master in the receiveEvent
String comando2;        // String to store the information sent by the master in the receiveEvent

int pin = A3;           // Arduino Pin A3 defined for the input signal
unsigned long val;      // Variable to store the measured signal
float vol;              // Variable to store the corresponding voltage
float peso;             // Variable to store the weight measurement
int cont;

// ************************* SETUP ******************************

void setup() {

  Wire.begin(Wdir);                   // Join I2C bus with address Wdir
  Wire.onRequest(requestEvent);    // It registers the function "requestEvent" that will be called by the master to
                                      // obtain information from the slave
  Wire.onReceive(receiveEvent);   // It registers the function "receiveEvent" that will be called when the slave
                                     // receives information
  Serial.begin(9600);                // Start the serial for the output information

  StrS.reserve(50);             // Reserve function allows the manipulation of the string StrS with a size of 50 bytes
  Rstr.reserve(100);            // Reserve function allows the manipulation of the string Rstr with a size of 100 bytes
  pinMode(pin, INPUT);      // Arduino pin A3 is defined as the input signal

}

// ************************* END SETUP ******************************

// ************************* LOOP ******************************

// Void loop stores in StrS string the characters read from the serial:

void loop() {

   if (Serial.available() > 0)
   // Serial.available provides the number of characters available to be read

    {
     ByteR = (char)Serial.read();      // Serial read reads the data available and char transforms it into char data type
     StrS += ByteR;
    }
```

```
}

// ************************** END LOOP ******************************

// ************************** SUB REQUEST_EVENT ******************************

// It transforms the Rstr string into char to be written in the I2C bus that will send information from the slave to the
// master

void requestEvent() {

  Rstr.toCharArray(cadena, Rstr.length() + 1);       // The string Rstr is copied into the char cadena
  Wire.write(cadena);                                // Writes the data in cadena
  Serial.println(cadena);
  Rstr = "";

}

// ************************** END REQUEST_EVENT **********************************

// ************************** SUB RECEIVE_EVENT ******************************

// It receives de information sent by the master

void receiveEvent(int howMany) {   // howMany defines the number of bytes read by the slave

  while (Wire.available())
  // While information is being sent the loop is performed
  // Stores the character read in c and adds it to the string comando

   {
    char c = Wire.read();
    comando = comando + c;
   }

  comando2 = comando.substring(0,4);    // comando2 stores characters from 0 (inclusive) to 4 (exclusive)
  StrS = comando;                       // comando is stored as StrS and leeStr is called
  leeStr();
  comando = "";

}

// ************************** END RECEIVE_EVENT **********************************

// ************************** SUB LEESTR ******************************

void leeStr(){

 if ( StrS[0]== 'S' && StrS[1]== 'W' && StrS[2]=='R') {  // If the command sent corresponds with "SWR":

    val = 0;
    cont = 0;

    for (cont = 0; cont<50; cont++)       // It gets 50 samples measured by the sensor
     {
      val = val + analogRead(A3);
     }


    val = val/50;                         // Calculates the average
    vol= float(val)*5.0/1024.0;           // Converts the value into voltage
    peso = vol*2.0;                       // Converts the voltage into weight using the calibration curve
```

```
    // As peso is a float variable it has to be converted into a char (dtostrf) and then into a string
    // There are four possibilities in this case, as the number of digits of the variable increases if the
    // value is equal or higher than 10, 100 and 1000, respectively:

    if (peso >= 0 && peso < 10 )
      {
       dtostrf(peso, 4, 2, RstrP1);
       Rstr = "&SWR*" + String(RstrP1) + "*";
      }

    if (peso >= 10 && peso <100)
      {
       dtostrf(peso, 5, 2, RstrP2);
       Rstr = "&SWR*" + String(RstrP2) + "*";
      }

    if (peso >= 100 && peso <1000)
      {
       dtostrf(peso, 6, 2, RstrP3);
       Rstr = "&SWR*" + String(RstrP3) + "*";
      }

    if (peso >= 1000 && peso <10000)
      {
       dtostrf(peso, 7, 2, RstrP4);
       Rstr = "&SWR*" + String(RstrP4) + "*";
      }
  }

}

// ************************* END LEESTR ******************************
```

9.6 pH-meter

```
// Arduino code to control the pH meter (A).

// There is a unique command pHAR that returns "&pHAR*phValue*" as a string to be read by Gambas.

#define SensorPin 0       // pH meter Analog output to Arduino Analog Input 0
// Define is used to name a constant without using memory space

#include <Wire.h>        // This line includes the library called Wire, which allows the communication with I2C
int Wdir = 16;           // Direction of the pH meter in the I2C bus

char ByteR;              // Byte read by the serial line to be stored in StrS

String Rstr = "";        // String to store output information
char RstrP1[4];          // Auxiliary variable to store output information
char RstrP2[5];          // Auxiliary variable to store output information
char cadena[21];         // Variable to store the output information in the requestEvent

String StrS = "";        // String to store input information
String comando;          // String to store the information sent by the master in the receiveEvent
String comando2;         // String to store the information sent by the master in the receiveEvent

int buf[10];                        // Variable to store values measured by the sensor
int temp;                           // Auxiliary variable to store values while sorted
```

```
unsigned long int avgValue;         // Store the average value of the sensor feedback
float phValue;                      // Stored pH value

// ************************** SETUP ********************************

void setup(){

  Wire.begin(Wdir);                   // Join I2C bus with address Wdir
  Wire.onRequest(requestEvent);    // It registers the function "requestEvent" that will be called by the master to
                                      // obtain information from the slave
  Wire.onReceive(receiveEvent);    // It registers the function "receiveEvent" that will be called when the slave
                                      // receives information

  Serial.begin(9600);          // Start the serial for the output information
  StrS.reserve(50);             // Reserve function allows the manipulation of the string StrS with a size of 50 bytes
  Rstr.reserve(100);           // Reserve function allows the manipulation of the string Rstr with a size of 100 bytes
  pinMode(13,OUTPUT);

}

// ************************** END SETUP ********************************

// ************************** LOOP ********************************

// Void loop stores in StrS string the characters read from the serial:

void loop() {

   if (Serial.available() > 0)
   // Serial.available provides the number of characters available to be read

    {
     ByteR = (char)Serial.read();   // "Serial.read" reads the data available and char transforms it into char data type
     StrS += ByteR;
    }

}

// ************************** END LOOP ********************************

// ************************** SUB REQUEST_EVENT ********************************

// It transforms the Rstr string into char to be written in the I2C bus that will send information from the slave to the
// master

void requestEvent() {

  Rstr.toCharArray(cadena, Rstr.length() + 1);   // The string Rstr is copied into the char cadena
  Wire.write(cadena);                                 // Writes the data in cadena
  Serial.println(cadena);
  Rstr = "";

}

// ************************** END REQUEST_EVENT ********************************

// ************************** SUB RECEIVE_EVENT ********************************

// It receives the information sent by the master

void receiveEvent(int howMany) {   // howMany defines the number of bytes read by the slave
```

```
  while (Wire.available())
  // While information is being sent the loop is performed
  // Stores the character read in c and adds it to the string comando

    {
     char c = Wire.read();
     comando = comando + c;
    }

  comando2 = comando.substring(0,4);    // comando2 stores characters from 0 (inclusive) to 4 (exclusive)
  StrS = comando;                       // comando is stored as StrS and leeStr is called
  leeStr();
  comando = "";

}

// ************************** END RECEIVE_EVENT ************************************

// ************************** SUB LEESTR *******************************

void leeStr(){

       if ( StrS[0]== 'p' && StrS[1]== 'H'  && StrS[2]== 'A'  && StrS[3]== 'R') {
      // If the command sent by the master corresponds with "pHAR":

              for(int i=0;i<10;i++)                 // Get 10 sample values from the sensor and store them in buf with
                                                    // a delay of 10 milliseconds
                {
                   buf[i]=analogRead(SensorPin);
                   delay(10);
                }

              for(int i=0;i<9;i++)       // Sorts the analog from small to large
                {
                   for(int j=i+1;j<10;j++)
                     {
                        if(buf[i]>buf[j])
                        // Compares two consecutive positions. If the first one is bigger the values are interchanged

                            {
                               temp=buf[i];
                               buf[i]=buf[j];
                               buf[j]=temp;
                            }
                     }
                }

              avgValue = 0;

              // Next, the average of the six center samples is calculated and converted from the
             // analog read into the pH value

              for(int i=2;i<8;i++)
              avgValue+=buf[i];

              phValue = (float)avgValue*5.0/1024/6;       // Converts the analog into millivolt
              phValue = 3.5*phValue;                      // Converts the millivolt into pH value

              Serial.print("pH:");
              Serial.print(phValue, 2);
              Serial.println(" ");
```

```
            digitalWrite(13, HIGH);
            delay(800);
            digitalWrite(13, LOW);


            // As phValue is a float variable it has to be converted into a char (dtostrf) and then into a string
            // There are two possibilities in this case, as the number of digits of the variable increases if the
            // value is equal or higher than 10:

            if (phValue < 10 )
              {
                dtostrf(phValue, 4, 2, RstrP1);
                Rstr = "&pHAR*" + String(RstrP1) + "*";
              }

            if (phValue >= 10 )
              {
                dtostrf(phValue, 5, 2, RstrP2);
                Rstr = "&pHAR*" + String(RstrP2) + "*";
              }
        }

}

// ************************* END LEESTR ******************************
```

10. Bibliography and references

1. WHO | Transplantation. *Whoint*. 2017. Available at: http://www.who.int/topics/transplantation/en/. Accessed August 22, 2017.

2. Orlando G, Wood K, Stratta R, Yoo J, Atala A, Soker S. Regenerative Medicine and Organ Transplantation: Past, Present, and Future. Transplantation. 2011;91(12):1310-1317. doi:10.1097/tp.0b013e318219ebb5.

3. Atala A. Printing a human kidney. Tedcom. 2011. Available at: https://www.ted.com/talks/anthony_atala_printing_a_human_kidney. Accessed August 22, 2017.

4. Atala A. Growing new organs. Tedcom. 2009. Available at: https://www.ted.com/talks/anthony_atala_growing_organs_engineering_tissue?language=gl. Accessed August 22, 2017.

5. Meirelles Júnior R, Salvalaggio P, Rezende M et al. Liver transplantation: history, outcomes and perspectives. *Einstein (São Paulo)*. 2015;13(1):149-152. doi:10.1590/s1679-45082015rw3164.

6. Baptista P, Orlando G, Mirmalek-Sani S, Siddiqui M, Atala A, Soker S. Whole organ decellularization - a tool for bioscaffold fabrication and organ bioengineering. 2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society. 2009. doi:10.1109/iembs.2009.5333145

7. Atala A, Bauer S, Soker S, Yoo J, Retik A. Tissue-engineered autologous bladders for patients needing cystoplasty. The Lancet. 2006;367(9518):1241-1246. doi:10.1016/s0140-6736(06)68438-9.

8. Sabetkish S, Kajbafzadeh A, Sabetkish N et al. Whole-organ tissue engineering: Decellularization and recellularization of three-dimensional matrix liver scaffolds. Journal of Biomedical Materials Research Part A. 2014;103(4):1498-1508. doi:10.1002/jbm.a.35291.

9. Chan S, Fan S, Lo C et al. A Decade of Right Liver Adult-to-Adult Living Donor Liver Transplantation. Transactions of the Meeting of the American Surgical Association. 2008;126:61-69. doi:10.1097/sla.0b013e31818584e6.

10. Shupe T, Williams M, Brown A, Willenberg B, Petersen BE. Method for the decellularization of intact rat liver. Organogenesis. 2010;6(2):134-136.

11. Uygun B, Soto-Gutierrez A, Yagi H et al. Organ reengineering through development of a transplantable recellularized liver graft using decellularized liver matrix. Nature Medicine. 2010;16(7):814-820. doi:10.1038/nm.2170.

12. Lechler R, Sykes M, Thomson A, Turka L. Organ transplantation—how much of the promise has been realized?. Nature Medicine. 2005;11(6):605-613. doi:10.1038/nm1251.

13. Arenas-Herrera J, Ko I, Atala A, Yoo J. Decellularization for whole organ bioengineering. Biomedical Materials. 2013;8(1):014106. doi:10.1088/1748-6041/8/1/014106.

14. Saidi RF, Hejazii Kenari SK. Challenges of Organ Shortage for Transplantation: Solutions and Opportunities. International Journal of Organ Transplantation Medicine. 2014;5(3):87-96.

15. Cubo N, Garcia M, del Cañizo J, Velasco D, Jorcano J. 3D bioprinting of functional human skin: production and in vivo analysis. *Biofabrication*. 2016;9(1):015006. doi:10.1088/1758-5090/9/1/015006.

16. Kang H, Lee S, Ko I, Kengla C, Yoo J, Atala A. A 3D bioprinting system to produce human-scale tissue constructs with structural integrity. *Nature Biotechnology*. 2016;34(3):312-319. doi:10.1038/nbt.3413.

17. Orlando G, Baptista P, Birchall M et al. Regenerative medicine as applied to solid organ transplantation: current status and future challenges. Transplant International. 2010;24(3):223-232. doi:10.1111/j.1432-2277.2010.01182.x.

18. Atala A. Tissue Engineering and Regenerative Medicine: Concepts for Clinical Application. Rejuvenation Research. 2004;7(1):15-31. doi:10.1089/154916804323105053.

19. Pellegata A, Dominioni T, Ballo F et al. Arterial Decellularized Scaffolds Produced Using an Innovative Automatic System. Cells Tissues Organs. 2015;200(6):363-373. doi:10.1159/000439082.

20. IUPAC. Compendium of Chemical Terminology. IUPAC Compendium of Chemical Terminology (2014). doi:10.1351/goldbook.I03352

21. Uzarski J, Bijonowski B, Wang B et al. Dual-Purpose Bioreactors to Monitor Noninvasive Physical and Biochemical Markers of Kidney and Liver Scaffold Recellularization. *Tissue Engineering Part C: Methods*. 2015;21(10):1032-1043. doi:10.1089/ten.tec.2014.0665.

22. Groth CG. The potential advantages of transplanting organs from pig to man: A transplant Surgeon's view. Indian Journal of Urology : IJU : Journal of the Urological Society of India. 2007;23(3):305-309. doi:10.4103/0970-1591.33729.

23. Taniguchi S, Cooper DK. Clinical xenotransplantation: past, present and future. Annals of The Royal College of Surgeons of England. 1997;79(1):13-19.

24. Abdel-Misih SRZ, Bloomston M. Liver Anatomy. The Surgical clinics of North America. 2010;90(4):643-653. doi:10.1016/j.suc.2010.04.017.

25. Tortora G, Derrickson B. Principles Of Anatomy & Physiology. 13th ed. Hoboken, NJ: Wiley; 2012:990-995.

26. Hall J, Guyton A. Guyton And Hall Textbook Of Medical Physiology. 13th ed. Philadelphia, PA: Elsevier; 2016:881-886.

27. Macchiarini P, Jungebluth P, Go T et al. Clinical transplantation of a tissue-engineered airway. *The Lancet*. 2008;372(9655):2023-2030. doi:10.1016/s0140-6736(08)61598-6.

28. Sullivan D, Mirmalek-Sani S, Deegan D et al. Decellularization methods of porcine kidneys for whole organ engineering using a high-throughput system. Biomaterials. 2012;33(31):7756-7764. doi:10.1016/j.biomaterials. 2012.07.023.

29. Song JJ, Guyette J, Gilpin S, Gonzalez G, Vacanti JP, Ott HC. Regeneration and Experimental Orthotopic Transplantation of a Bioengineered Kidney. Nature medicine. 2013;19(5):646-651. doi:10.1038/nm.3154.

30. Orlando G, Booth C, Wang Z et al. Discarded human kidneys as a source of ECM scaffold for kidney regeneration technologies. Biomaterials. 2013;34(24):5915-5925. doi:10.1016/j.biomaterials.2013.04.033.

31. Ott H, Matthiesen T, Goh S et al. Perfusion-decellularized matrix: using nature's platform to engineer a bioartificial heart. Nature Medicine. 2008;14(2):213-221. doi:10.1038/nm1684.

32. Ko I, Peng L, Peloso A et al. Bioengineered transplantable porcine livers with re-endothelialized vasculature. *Biomaterials*. 2015;40:72-79. doi:10.1016/j.biomaterials. 2014.11.027.

33. Gambas - Gambas Almost Means Basic. Gambassourceforgenet. 2017. Available at: http://gambas.sourceforge.net/en/main.html#. Accessed August 27, 2017.

34. Bus I, That? I. I2C - What's That? - I2C Bus. I2C Bus. 2017. Available at: https://www.i2c-bus.org/. Accessed August 27, 2017.

35. Arduino - Wire. Arduinocc. 2017. Available at: https://www.arduino.cc/en/Reference/Wire. Accessed August 27, 2017.

36. European Medicines Agency -. Emaeuropaeu. 2017. Available at: http://www.ema.europa.eu/ema/. Accessed August 30, 2017.

37. Agencia Española de Medicamentos y Productos Sanitarios. 2017. Available at: https://www.aemps.gob.es/. Accessed August 30, 2017.

38. Soto-Gutierrez A, Wertheim J, Ott H, Gilbert T. Perspectives on whole-organ assembly: moving toward transplantation on demand. Journal of Clinical Investigation. 2012;122(11):3817-3823. doi:10.1172/jci61974.

39. Crapo P, Gilbert T, Badylak S. An overview of tissue and whole organ decellularization processes. Biomaterials. 2011;32(12):3233-3243. doi:10.1016/j.biomaterials.2011.01.057.

40. Baptista P, Siddiqui M, Lozier G, Rodriguez S, Atala A, Soker S. The use of whole organ decellularization for the generation of a vascularized liver organoid. Hepatology. 2011;53(2):604-617. doi:10.1002/hep.24067.

41. Ross E, Williams M, Hamazaki T et al. Embryonic Stem Cells Proliferate and Differentiate when Seeded into Kidney Scaffolds. Journal of the American Society of Nephrology. 2009;20(11):2338-2347. doi:10.1681/asn.2008111196.

42. Soto-Gutierrez A, Zhang L, Medberry C et al. A Whole-Organ Regenerative Medicine Approach for Liver Replacement. Tissue Engineering Part C: Methods. 2011;17(6):677-686. doi:10.1089/ten.tec.2010.0698.

43. Lawrence B, Devarapalli M, Madihally S. Flow dynamics in bioreactors containing tissue engineering scaffolds. Biotechnology and Bioengineering. 2009;102(3):935-947. doi:10.1002/bit.22106.

44. Ito T, Kiuchi T, Yamamoto H et al. Changes in portal venous pressure in the early phase after living donor liver transplantation: pathogenesis and clinical implications1,2. Transplantation. 2003;75(8):1313-1317. doi:10.1097/01.tp.0000063707.90525.10.

45. Lin P, Chan W, Badylak S, Bhatia S. Assessing Porcine Liver-Derived Biomatrix for Hepatic Tissue Engineering. Tissue Engineering. 2004;10(7-8):1046-1053. doi:10.1089/ten.2004.10.1046.

46. Badylak S. Xenogeneic extracellular matrix as a scaffold for tissue reconstruction. Transplant Immunology. 2004;12(3-4):367-377. doi:10.1016/j.trim.2003.12.016.

47. Chun S, Lim G, Kwon T et al. Identification and characterization of bioactive factors in bladder submucosa matrix. Biomaterials. 2007;28(29):4251-4256. doi:10.1016/j.biomaterials.2007.05.020.

48. Fukumitsu K, Yagi H, Soto-Gutierrez A. Bioengineering in Organ Transplantation: Targeting the Liver. Transplantation Proceedings. 2011;43(6):2137-2138. doi:10.1016/j.transproceed.2011.05.014.

49. Nathan H, Conrad S, Held P et al. Organ donation in the United States. American Journal of Transplantation. 2003;3(s4):29-40. doi:10.1034/j.1600-6143.3.s4.4.x.